Dependable IMS services - A Performance Analysis of Server Replication and Mid-Session Inter-Domain Handover

Renier, Thibault Julien

Publication date:
2008

Document Version
Publisher's PDF, also known as Version of record

Dependable IMS Services
–
A Performance Analysis of Server Replication and Mid-Session Inter-Domain Handover

Ph.D. Dissertation

by

Thibault Renier

June 2008

Center for TeleInfrastruktur
Department of Electronic Systems
Aalborg University
Aalborg, Denmark

Supervisors

Prof., Dr. Ramjee Prasad
Aalborg University, Denmark

Assoc. Prof., Dr. Hans-Peter Schwefel
Aalborg University, Denmark

Assessment Committee

Assoc. Prof., Henrik Schiøler (Chairman)
Aalborg University, Denmark

Dr. Hendrik Berndt
Docomo Europe, Munich, Germany

Assoc. Prof., Dr. Andrea Bondavalli
University of Florence, Italy

Moderator

Prof. Ole Brun Madsen
Aalborg University, Denmark

ISSN 0908 – 1224

ISBN 87 – 90834 – XX – XX

Abstract

Dependability has been extensively investigated in the industry for decades. Now that telecommunications markets are getting more competitive than ever, making telecommunications networks dependable has become another critical commercial argument for operators. At the same time, telecommunication networks are converging towards the all-IP paradigm. Despite the undeniable advantages brought by IP-based architectures, the openness of the IP paradigm has forced operators to devise control mechanisms such as access and service control in order to limit the access to their resources to authorized users only. The IP Multimedia Subsystem (IMS) is an access technology-independent implementation of IP-based (1) access, (2) service and (3) session control platform. The IMS was originally designed by 3GPP for mobile access networks, namely UMTS, but its specifications were extended so that it can be deployed with any IP-based access network and also support communications with non-IP networks like PSTN telco networks.
Such control platforms add complexity and introduce new points of failures into the systems they are deployed with. Potentially poor dependability levels in the IMS should not cancel the considerable efforts made in the area of dependable end-user applications. It is therefore crucial that faults occurring in the IMS do not impact the overall dependability levels of the system, which means that the IMS needs to be at least as dependable as the applications it controls. Meanwhile, the IMS performance as perceived by the end-user should not be degraded either, since it is undeniably an important criterion for user satisfaction. Fault tolerance mechanisms tend to decrease performance; hence, it is a real challenge to improve both dependability and performance of a service simultaneously. The goal of this thesis is to analyze the performance capabilities of dependable, IMS-controlled UMTS access networks and suggest optimizations of the fault tolerance mechanisms in order to control the dependability/performance tradeoff to meet given sets of dependability and performance requirements. The main fault tolerance solutions considered to raise the IMS dependability are (1) request retransmissions to cope with temporary network faults and (2) IMS server replication to support failover when a server is no longer reachable.

Both the standard and replicated IMS architectures are modeled with Stochastic Activity Networks in order to evaluate the tradeoff between dependability, performance (IMS service access time) and additional costs (overall IMS load). Some of the parameters of the failure detection and recovery mechanisms are tuned to analyze which parameters the tradeoff is more sensitive to. This way, an optimal fault tolerance configuration can be set in a given system with specific requirements. Optimal configuration selection strategies (at design time and at run time) are discussed and a selection example illustrates how to use the simulation results to do so.
Most IMS servers are distributed stateful entities that keep track of several states (e.g., session state, charging state). Thus, state inconsistency can affect dependability when the traffic is switched from one server to another during failovers or to support a new transaction (server selection policies). Existing techniques propose to control the tradeoff between performance and dependability by dynamically delaying the processing of a service request accordingly to the current inconsistency level. The efficiency of such

algorithms is greatly dependent on the accuracy of the inconsistency metric that they use as input. Therefore, a new inconsistency evaluation framework for Internet-like services is introduced and validated experimentally. This framework can be used at system design time and run time in order to evaluate the probability of reading a stale state replica at a given server.
Finally, when non-replicated components of a system fail, failovers and retransmissions to the replicated IMS servers cannot overcome the faults occurring at these non-replicated components. Then, the best strategy for a client is to switch to another access network, which requires macro mobility support. While standard mobility solutions support intra-domain handovers, the IMS specifications do not allow for mid-session inter-domain handover (i.e. the IP address of the user equipment changes during ongoing sessions) yet. In this thesis, a Mobile IP-based solution for mid-session macro mobility is investigated in two steps. First, interoperability problems between Mobile IP and SIP operations are highlighted and the adequate protocol and functionality adjustments are proposed to make the deployment of Mobile IP possible in IMS-controlled access networks. Then, Mobile IP-based handover optimizations are introduced that dramatically decrease the mid-session inter-domain handover time. The handover times are evaluated analytically for several scenarios of network delays and number of media streams in the moved IMS session. Finally, it is analytically analyzed under which conditions—on network delays and number of media streams—the SIP operations are faster than the novel enhanced MIP-based solution. This analysis proves that the SIP solution can perform better than MIP in a very few cases.

The original contributions of the thesis are the following:

- SAN models and Möbius implementations of the standard IMS and replicated IMS scenarios
- Parametric analysis of relevant input system-state variables and fault tolerance configuration settings
- Strategies for the selection of the optimal fault tolerance configuration
- Novel inconsistency evaluation framework
- Strategies for network architecture design based on inconsistency/performance requirements
- MIP and IMS function/protocol extensions for MIP-based network layer macro mobility in IMS environments—to support session continuity
- Novel enhanced MIP-based macro handover

Dansk Resumé

Acknowledgments

Table of Contents

List of Figures

List of Tables

List of Acronyms

2G	Second Generation mobile systems
3G	Third Generation mobile networks
3GPP	Third Partnership Project
AKA	Authentication Key Agreement
AN	Access Network
AN_{new}	AN a UE is attached to *after* a handover
AN_{old}	AN a UE is attached to *before* a handover
API	Application Programming Interface
AR	Access Router
ARQ	Automatic Repeat reQuest
AS	Application Server
ASAP	Aggregate Server Access Protocol
B3G	Beyond 3G
BS	Bearer Service
BU	Binding Update
CDMA	Code Division Multiple Access
CF	Contributing Factor
CL	Cycle Length
CN	Corresponding Node
CoA	Care-of-Address
COPS	Common Open Policy Service
COTS	Commercial-Off-The-Shelf
CRC	Cyclic Redundancy Check
CSeq	Sequence number
CSCF	Call Session Control Function
DCCP	Datagram Congestion Control Protocol
DoS	Denial of Service
DP	Dissemination Protocol
E2E	End-to-End
ENRP	Endpoint Handlespace Redundancy Protocol *(previously called* Endpoint Name Resolution Protocol*)*
FDM	Failure Detection Mechanism
FEC	Forward Error Correction
FM	Failover Mechanism
FMC	Fixed-Mobile Convergence
FO	Failover

GGSN	Gateway GPRS Support Node
GPRS	General Packet Radio Service
GSM	Global System for Mobile communications
HA	Home Agent
HARQ	Hybrid ARQ
HB	Heartbeat
HBA	Hash Based Addresses
HIP	Host Identity Protocol
HoA	Home Address
HSS	Home Subscriber Server
HTTP	Hyper-Text Transfer Protocol
I-CSCF	Interrogating CSCF
IETF	Internet Engineering Task Force
IMS	IP Multimedia Subsystem
InterSIP/HB	Inter-SIP/HB time
IP	Internet Protocol
ISDN	Integrated Services Digital Network
LAN	Local Area Network
LIFO	Last In, First Out
M-MIP	Multihomed MIP
MGW	Media GateWay
MGCF	Media Gateway Control Function
MIP	Mobile IP
MN	Mobile Node
MPLS	Multi-Protocol Label Switching
Msg	message
MSH	Mid-Session Handover
NS	Name Server
P_{ON}/P_{OFF}	Probability that a server is ON or OFF, respectively
P-CSCF	Proxy CSCF
PCF	Policy Control Function
PDP	Packet Data Protocol
PE	Pool Element
PEP	Policy Enforcement Point
PER	Packet Error Rate
PS-CN	Packet-Switched Core Network
PSTN	Public Switched Telephone Network
PU	Pool User
QoS	Quality of Service

RAN	Radio Access Network
RFC	Request For Comments
RO	Read Operation
RSerPool	Reliable Server Pooling
RSVP	Resource reSerVation Protocol
RTCP	RTP Control Protocol
RTP	Reliable Telco Platform
RTT	Round Trip Time
S-CSCF	Serving CSCF
SACK	Selective ACKnowledgment
SAN	Stochastic Activity Network
SAT	Service Access Time
SBLP	Service-Based Local Policy
SCTP	Stream Control Transmission Protocol
SDP	Session Description Protocol
SGSN	Service GPRS Support Node
SIP	Session Initiation Protocol
SLA	Service Level Agreements
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SOM	State Ordering Metric
SPI	IPSec Security Parameter Index
SS7	Signaling System 7
SSA	State Sharing Algorithm
SSP	Server Selection Policy
SUM	State Update Message
TCP	Transmission Control Protocol
TFT	Traffic Flow Template
ToS	Theft of Service
TSN	Transport Sequence Number
TTF	Time To Failure
TTR	Time To Repair
UAC/UAS	User Agent Client/Server
UDP	User Datagram Protocol
UE	User Equipment
UMTS	Universal Mobile Telecommunication System
URI/URL	Uniform Resource Identifier/Locator
USRR	Unsuccessful State Replication Rate
UTRAN	UMTS Terrestrial RAN
VoIP	Voice over IP
WLAN	Wireless Local Area Network

1. Introduction & Problem Definition

1.1. Background

After more than a decade of operation, second generation (2G) mobile systems such as the Global System for Mobile communications (GSM) have eventually reached their limits in terms of penetration rates and average revenue per user. Based on circuit-switched technologies, 2G systems provide the users with wireless access to voice communications and a restricted set of data services such as short message service (SMS) and fax. In order to increase revenues, mobile operators had to plan the evolution of their aging 2G infrastructures and services.
Until then, the jump to a new generation of mobile network was characterized by the deployment of a new access technology that would offer higher wireless access bandwidth and capacity to cope with the constantly rising number of subscribers and the more complex network management and operations. Such upgrades typically improved availability and quality of service (QoS). As availability and QoS have became consistently satisfactory, users start taking these two essential aspects of personal mobile communications for granted and now expect more in terms of range of applications, performance, and ease of use. Thus, services are becoming one of the key factors to create new revenues for mobile operators.

Driven by this new demand from the users, system designers introduced a critical change in 2.5G systems (e.g., General Packet Radio Service (GPRS) [3GPP04b]) as compared to 2G systems: the transport of data in the core network of these systems is IP-based (i.e. packet-based). The change to IP-based transport has been further extended to the radio access network (RAN) from 3G systems, whose deployment examples include the Universal Mobile Telecommunications Systems (UMTS) [3GPP04a].
In 2G systems, services were proprietary solutions and were solely managed and upgraded by the mobile operators. By building on top of the IP principles and the Internet for 2.5G and beyond, (1) applications are not bound to a legacy access technology anymore and, thus, they can be maintained even as the underlying access networks change and evolve and (2) independent parties can also provide services to mobile systems. The shift to IP introduces a flexible platform for service development and deployment, and encourages the provisioning of innovative mobile multimedia services. Also, many of the existing services available in the fixed IP world can be quickly adapted to fit the mobile devices capabilities and, then, be offered to mobile users. Finally, by allowing multiple network technologies to coexist, the IP paradigm is able to provide ubiquitous connectivity and to achieve enormous economies of scale.

The support of IP-based services—especially real-time services—raises new challenges for the mobile operators though.

IP is a connectionless protocol that transmits its packets on a best effort basis; therefore, QoS is not supported by default in IP-based systems. On the other hand, real-time services have strict performance requirements in order to meet the user demands in terms of QoS. One solution is to deploy additional mechanisms in order to organize multimedia communications into sessions that can be managed individually. A session consists in (1) separating the communications of a given multimedia application into independent IP data streams—e.g. for a videoconference: one stream for the voice and one stream for the video—and (2) setting the expected end-to-end (E2E) QoS levels for each stream, which requires special QoS support in the operators' networks (e.g. PDP contexts associated to TFT and SBLP filters, see Section 2.3.2) and in the networks interconnecting remote access networks (e.g. MPLS [Rosen01], RSVP [Braden97]). Relevant examples of IP-based sessions are Internet phone calls and instant messaging (e.g. Skype, MSN messenger), multi-party multimedia conferences and online gaming.
Users are getting used to accessing the Internet on the move thanks to the success of novel wireless access technologies such as WLAN [IEEE_web] and Bluetooth [Bluetooth_web]. These access technologies were originally designed for high bandwidth and short-range communications in hot spots and home/personal networks [MAGNET_web][Prasad06]. Also, with the promise that IP-based systems will be the basis for cross-network service provisioning, users should have the possibility to connect to a variety of wireless access networks on-demand in order to be "always best connected" to the services of their choice. The possibility to switch between access technologies will (1) extend the coverage of the traditional mobile cellular systems in low density areas where it would be too expensive to deploy cellular equipment, and (2) allow to select the most appropriate access network to meet the users' preferences (e.g. cost, QoS). Therefore, inter-domain mobility, so-called macro mobility, is a critical challenge to address when designing 3G and beyond 3G (B3G) systems. With users frequently switching from one access network to another, mobile operators have to control the access to their scarce radio resources by deploying:

- Access and service controls, to block unauthorized users, prevent theft of services,
- session control, to map users' subscription profiles to authorized QoS levels in the access network.

The standardization bodies took those important aspects of 3G and B3G systems into account and specified an IP-based platform for access, service and session control in cellular settings—the so-called IP Multimedia Subsystem (IMS), which was introduced by 3GPP in Release 5 of the UMTS specifications [3GPP02a][3GPP02b]. Even though the IMS was originally designed exclusively for UMTS networks, 3GPP quickly made it network-independent by designing a set of required interfaces between the IMS platform and the access network it should control. IMS procedures rely on modified versions of SIP [Rosenberg02] mechanisms and servers. The IMS is now considered as the platform of choice for providing unified session control on top of multiple access technologies and for supporting flexible multimedia applications.

In summary, the need for a wider diversity of services accessible from mobile devices has encouraged the transition to IP-based mobile networks, as IP seems to be an excellent paradigm for fast and flexible development and deployment of mobile services. In

parallel, the convergence of several access technologies will considerably improve the user's experience. Therefore, roaming between different access technologies—even during ongoing sessions—is becoming one of the main challenges considered in the definition of 3G and B3G systems. Ultimately, the goal is to provide users with any content, anytime, anywhere.
To make this possible, a major step is the deployment of the IMS, which acts as the common platform for access/service control and multimedia session control in IP-based networks.

1.2. Problem Statement

As discussed in the previous section, the deployment of new services and support for macro mobility are essential characteristics of upcoming mobile IP-based networks. Nonetheless, the quality and the dependability of multimedia sessions remain constant requirements from users that mobile operators must meet consistently for fear of losing market shares. For years, considerable efforts have been made in order to raise the dependability levels in communications networks, which lead in turn to the dramatic improvement of the availability and reliability of end-user services and network operations. This evolution was made possible by developing stable software (e.g., systematic testing, efficient programming languages, and software and operating systems design), robust hardware (e.g., better materials and architectural designs), and dependability solutions (e.g., redundant subsystems). Nowadays, users expect that almost every request to access a mobile service is successful and the required QoS for that service can be sustained QoS during the whole service provisioning. In other terms, end-user services should be available at any time and, once initiated, multimedia sessions should not be interrupted due to faults occurring in the system (cf. Section 1.3 for the fault model and other limitations chosen in this report) or degraded QoS.
The IMS relies on extensive procedures involving multiple entities, which adds complexity and introduces new points of failures into the overall system. If SIP requests are lost during the session initiation procedures, the session setup delays can significantly increase [Fathi06] and some sessions may not even be initiated at all, which means that the SIP service is unavailable. Potentially poor dependability levels in the IMS should not cancel the productive efforts made in the area of dependable end-user services and network operations. It is therefore crucial that faults occurring in the IMS do not impact the dependability levels of the overall system; so, the IMS needs to be at least as dependable as the applications it controls.
3GPP is only responsible for defining how the IMS interworks with access networks—and application platforms—through standardized interfaces. In the standard IMS setup, fault tolerance mechanisms are limited to the SIP timeouts-per-request and request retransmissions; no IMS-specific dependability solution has been specified by 3GPP. Thus, it is necessary to consider additional techniques that can improve the IMS dependability.
The integration of dependability solutions in IMS-controlled systems will raise the IMS dependability levels closer to those already achieved by some dependable end-user services. Unfortunately, the deployment of dependability solutions implies a tradeoff between dependability and performance [Heddaya96] so, when addressing fault

scenarios, performance needs to be addressed in relation to dependability so that the mechanisms that make the IMS dependable should impact its performance as little as possible.

In this work, the means considered for improving IMS dependability is redundancy. Server replication is a popular approach to mask node failures due to local hardware and/or software faults, as well as network failures caused by interferences or packet losses [Helal96][Tanenbaum02]. Replicating some parts of a system seems especially suited for the dependability/performance tradeoff optimization challenge as it is sometimes used to increase the capacity— and therefore increase performance as well— in heavily loaded systems by offering more resources, whether it be bandwidth, processing power, memory, etc.
The overall goal of this thesis is to investigate the dependability/performance tradeoff in IMS-controlled UMTS systems, when using replication techniques in order to support dependable and high-performance service provisioning in the following failure+recovery scenarios:

- **Part I—IMS Server Replication.** Like any other components of the UMTS system, IMS servers are expected to fail or to be isolated from other entities because of network failures. For coping with temporary network failures (e.g., a random packet loss) the solution is to 'replicate' the original unsuccessful request and retransmit it to the server. If the retransmitted requests are also unsuccessful, either because the network failures are not temporary (e.g., a cable breakage that is not detected quickly) or because the server contacted is crashed, an interesting alternative to request retransmissions is to replicate the IMS servers. Then, the ongoing sessions supported by a failed server can be moved altogether to one backup server, or split among multiple server replicas, in order to mask the server failure and maintain the IMS service provisioning—this is called failover. The right combination of retransmissions to one server and number of failovers needs to be looked into to optimize the dependability/performance tradeoff.
  Failover procedures rely on the client side; server-local mechanisms such as hardware, operating system, or application restarts exist [vanMoorsel06] but they are not investigated in this work.
- **Part II—Mid-Session Macro Handover.** It is undeniable that replicating servers is an effective way to raise dependability; on the other hand, it can be costly to deploy multiple servers because of the large expenses this entails (server hardware & software, wiring, network configuration, etc.) and because of the increased overhead in the system. Therefore, it is rarely possible to replicate all the components that would potentially crash.
  A *single point of failure* is a system component that is not replicated and that prevents any level of service provisioning when it has failed. In the IMS-controlled UMTS scenario, the UMTS access router is an example of critical entity that prevents any traffic between a user equipment attached to this UMTS network and external entities, such as an application server or even an IMS server. In this scenario, retransmissions are ineffective when the component is permanently faulty or faulty long enough to consider another recovery solution. A good option for the user equipment is then to connect to another system, where hopefully no single point of

failure is currently crashed. Hence, macro mobility between IMS-controlled UMTS networks is investigated as a means for fault tolerance against single points of failures in the access network or in the IMS. In a way, this scenario is a specific case of replication where the access networks would be the replicated component of a 'global' system.

1.3. Terminology and Problem Limitation

In this section, the main dependability- and network environment-related terms that will be used throughout the report are defined. The scope of the problem, the level of details and the assumptions for the investigation are also specified for both dependability and the network environment.

1.3.1. Dependability

Terminology

According to [Avizienis04], a function is defined as what a subsystem is intended to do, which is described in the subsystem specifications in terms of functionality and performance. A service is the functionality provided by a component, as seen by its user. An *error* is a deviation of the function behavior from its functional specifications. *Faults* are what cause errors and they can be verified or hypothesized (e.g., 'is the remote process currently crashed or slow?'). In general, systems are vulnerable to faults of varied nature: hardware component crashes, incorrect software behavior, system overload, human errors, physically damaged wires and connectors, etc. In communication networks, faults can be classified into two distinct high-level classes:

- Node fault: when a node stops providing the expected service(s),
- Link fault: when a link stops serving as a medium to convey packets.

When an error becomes perceivable by the users (or subsystems) that requested the faulty service, it is called *failure*. Sometimes, the service is provided in a degraded mode, and the failure is said to be partial; when the user does not have any access to the service, the failure is total.

Ideally, *dependability* is the ability of a system to avoid, or hide, any failure that would prevent the service to keep behaving as expected, i.e. as written in the specifications. In practice, dependability most often consists in "avoiding service failures that are more frequent and/or more severe than is acceptable" [Avizienis04]. Dependability has the following attributes:

- *Availability* relates to readiness for correct service, i.e. the ability to start the service execution when the requests are received by a system;
- *Reliability* characterizes the ability to continuously provide correct service execution.
- *Security* is the level of trust that can be put in the exchanged information. Security is a composite of confidentiality (the information cannot be intercepted by third-parties) and integrity (the information cannot be corrupted by third-parties).

These attributes sometimes rely on sub-attributes such as *consistency*, which reflects the ability to provide the system with a consistent image of the state information stored over multiple replicas.

Dependability can be achieved by implementing the following means:

- *Fault prevention*: to prevent the occurrence or introduction of faults.
- *Fault tolerance*: to avoid service failures in the presence of faults.
- *Fault removal*: to reduce the number and severity of faults.
- *Fault forecasting*: to estimate the present number and the likely consequences of faults. Sometimes, fault forecasting deals with predicting faults from the current and past system state/behavior. The difference with fault prevention is that fault prevention is achieved at the system design time, while fault forecasting is done during the system run time.

Fault Model

Dependability. The only attributes of dependability investigated are availability and reliability, which both relate to the readiness and continuity of correct service provisioning. Security is accessorily treated in the sense that the design of non-standard solutions proposed in this work should not weaken the overall system security, but the main intention here is not to investigate additional means to raise the security levels.

Dependability means. As motivated in the previous subsection, replication—or redundancy—is the means considered to achieve high dependability in IMS-controlled systems. Faults are not prevented, removed, or forecast; instead, the approach taken consists in avoiding or minimizing the effects of faulty components by switching over to backup resources when faults are suspected.

In replicated systems, the usual recovery strategy consists in isolating components suspected to be failed or unreachable and involving redundant resources to take over the ongoing tasks of the failed components. The detailed suite of mechanisms required to achieve fault tolerance in a replicated architecture are given in Section 3.2.1, and can be summarized as follows:

- *Fault Detection*: A faulty component must be detected as quickly as possible so that a backup component takes over rapidly and that the faulty component is removed from the system until it is repaired (or replaced). Fault diagnostic is the specific ability to determine the nature of the faults so as to invoke the most adequate recovery mechanism.
- *Failover*: A failover is the most common recovery mechanism in redundant systems. When a service cannot be delivered by a component, the system should stop using the latter and switch to a backup component for future service requests.

In Part I, SIP server crashes and network faults are detected using (1) a heartbeat mechanism involving an independent failure detector and (2) the standard SIP timeout-per-request. When the failure detector suspects SIP servers to be unavailable, clients stop contacting the potentially failed servers for future requests. Also, when a SIP client does not receive a request acknowledgement from a seemingly available server, the SIP request is retransmitted a given number of times. If the request acknowledgement has not been received before the maximum number of retransmissions is reached (seven

transmissions total in the standard SIP setting, see Section 3.1.4), the server is considered failed by the client, and the latter triggers a failover by contacting a backup SIP server instead.
In Part II, failure detection is outside the problem scope and the investigation assumes that the clients are aware of a crash at a single point of failure in the system. The analysis starts at the point when the client triggers the necessary recovery action, i.e. a handover from the current access network to another access network.

Faults and failures. In Part I, the following assumptions on faults and failures occurring in IMS-controlled UMTS systems are made, which provides a framework for the analysis:

- *Node faults*: Replicating all IMS servers would be highly relevant to provide optimal IMS service dependability in a real setting. Nevertheless, the analysis only focuses on the IMS Serving Call Session Control Function (S-CSCF) replication because:
  - the S-CSCF is the most critical function for IMS session control;
  - the load on the S-CSCF server is expected to be higher than on other IMS entities, which justifies to replicate the S-CSCF both for dependability and capacity purposes;
  - it is not likely that all IMS entities can be replicated, e.g. because of the overhead and budget limits;
  - the conclusions drawn from the present S-CSCF replication analysis can be applied to other IMS redundant server scenarios.

  The S-CSCF fault model is quite simple as it only consists of crash faults, i.e. the server stops responding to any request. The S-CSCF crash faults are not permanent; they are halt faults modeled by a two-state (ON/OFF) Markov model. This model abstracts the nature of the faults and assumes that each fault (e.g. hardware breakage, software instability) leads to a complete S-CSCF failure; no partial failure is considered here. Also, it is assumed that if any protocol layer of the S-CSCF acknowledges a request (e.g. heartbeat request at Layer 4), all the other functions in this node are also available. This simplifies the problem because the only diagnostic to be made is ‘is the node alive?’, instead of the more refined diagnostic ‘up to which layer is the node available?’ that is required in a real setting.
- *Link faults*: In general, faults occurring in the communication path are mainly due to physical link breakage, wireless link interferences leading to high bit error rate, or capacity shortage (e.g. router overload) and typically translate into (1) packets losses or (2) complete remote endpoint unreachability due to loss of coverage. In this work, all link failures are modeled by a unified packet loss probability.
- *Timing failures*: From the SIP application (i.e. service provisioning) point of view, the faults assumed here lead to timing failures. Timing failures occur when the response to a client request is not received before the request timeout expires. Note that communication and processing delays vary over time and may also lead to timing failures.

In Part II, since crashes at a single point of failure are assumed to be already detected and the analysis starts when the macro handover mechanisms are initiated, the nature and occurrence of the faults that cause the complete access network failure do not matter for the analysis. Additionally, the analysis of the macro mobility solution is done with the

assumption that no fault affects the new access network or any entity required to support mobility mechanisms during the whole handover procedure.

A detailed review of the specific faults considered for each contribution is given in the respective contribution chapters.

1.3.2. Network Topology & Service Provisioning

Terminology

In this subsection, the network topology-related terms are defined—and illustrated in Figure 1.1—as well as several other concepts behind service provisioning in an IMS-controlled access network:

The term *access network* (AN) usually refers to a radio access network, i.e. the part of a mobile system comprising the access interface (wireless or wired) and the fixed part of an operator domain between the access links and the core network. Here, AN corresponds to the whole operator domain, which consists of (1) the radio access network and (2) the core network, up to its *access router* (AR) of the operator domain. Note that one operator can deploy multiple domains. Also, despite the fact that UMTS access networks can connect to heterogeneous external networks (e.g. the Internet, private IP networks or a Public Switched Telephone Network (PSTN)), it is assumed that all traffic between a *user equipment* (UE) and any external entity is IP-based; so, all external networks are unvaryingly referred to as the *Internet*. Throughout the whole document, the terms access network, mobile network and operator's domain are interchangeably used to refer to the infrastructure between a UE and its current access router to the Internet. In UMTS, the AN infrastructure mostly consists of the *UMTS Terrestrial Radio Access Network* (UTRAN) and the UMTS, so-called *Packet-Switched Core Network* (PS-CN).

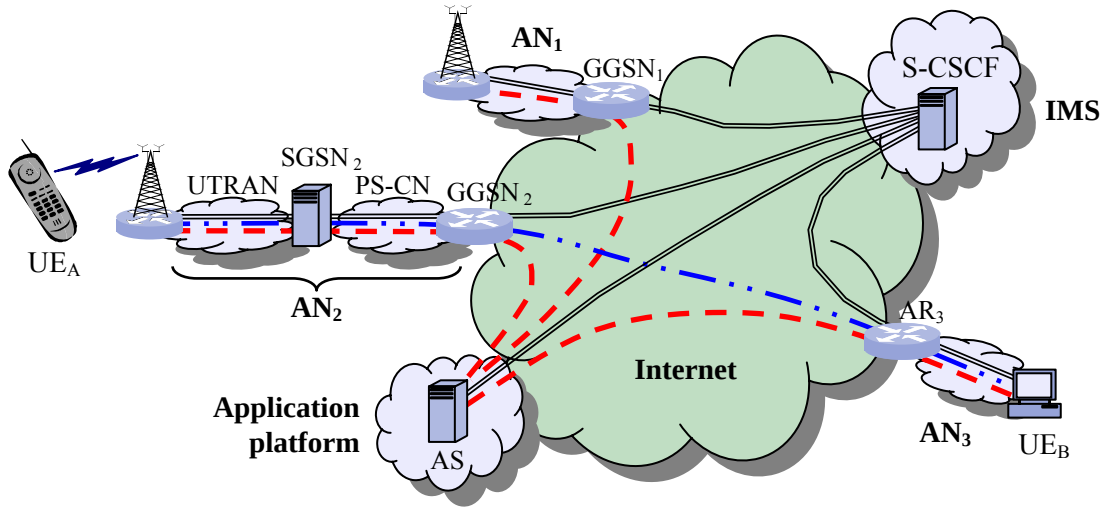


Fig. 1.1, High-level network architecture

Figure 1.1 shows a high-level network architecture for the example of three access networks (AN_1 and AN_2 are wireless, AN_3 is wired) controlled by a unique IMS (signaling traffic is represented by the black lines). This scenario corresponds to the case when an operator deploys multiple ANs around the Internet and centralizes the session control in one 'remote' IMS. The actual IMS architecture comprises more entities than just the S-CSCF and is fully described in Section 2.3.1. In the present example, the operator outsources the whole end-user service provisioning (e.g. web-pages, file downloads, emails) to an external third-party provider (red dashed lines). Two UEs (UE_A and UE_B) connected to different ANs communicate (e.g. VoIP phone call) directly via the Internet (dashed blue line), as the IMS is not involved in the data traffic. UE_A is connected to a UMTS AN, namely AN_2, for which the internal network architecture with the UTRAN and the PS-CN is shown; note that AN_1 is identical to AN_2 its architecture is not detailed.

One of the strengths of IMS-controlled access networks is that applications can be deployed outside of the operator domains, e.g. by third-party providers. An *application platform* corresponds to the network where a set of *application servers* (AS) is deployed and managed.
Access control is the procedure that grants access network connectivity to authorized users and block the others. This connectivity is usually given only within the bounds of the specific access network the UE attaches to and often only allows signaling traffic to and from the control functions of the AN. *Service control* mechanisms are initiated after successful access control and are responsible for authorizing a UE to access a requested service only if the user has subscribed to this service. Once a service has been authorized, *session control* solutions may set up a session by negotiating and enforcing some QoS session parameters as well as the corresponding charging scheme(s).
In UMTS, control functions are deployed in the control plane. The access control is executed during the GPRS Attach procedure, while both the service and session controls are handled by the IMS, which negotiates and reserves the QoS resources during the SIP session setup. The QoS policies resulting from the IMS negotiation phase are applied into the whole UMTS AN—i.e. between the UE and the GGSN—by reserving resources for each IP session flow, which is carried through an individual Packet Data Protocol (PDP) context (see Section 2.3.2).

Problem Scope

The goal of this work is to provide dependable IMS service provisioning, which relies on E2E multimedia session support. In this context, mechanisms at the IP and upper layers are the primary focus. Consequently, Layer 2 operations specific to legacy mobile networks are not of special interest, except for the QoS allocation and management functions in UMTS ANs that interact with the IMS via dedicated interfaces (mainly the GGSN).

1.4. Refined Problem Statement and Contributions

1.4.1. Part I - IMS Server Replication

In this study, (1) S-CSCF servers are replicated to provide fault tolerance when the active S-CSCF goes down and (2) SIP requests are retransmitted to the primary S-CSCF to recover from failures caused by random packet losses or temporary long communication delays. The replicated S-CSCFs are gathered in a logical group, which is managed by a protocol suite implemented at the middleware. Background information on the two main paradigms for server replication—a distributed approach and a cluster-based approach—is given in Section 3.2. The analysis is based on the distributed approach (implemented by the RSerPool protocol suite [Lei07]) but the main results are also discussed in terms of how using the cluster-based approach would affect them.

Optimal Fault Tolerance Configuration Selection

Server replication improves service dependability but it also affects performance and adds overhead and complexity, which also worsen performance. It is therefore essential to investigate how to minimize the performance degradation introduced by replicating servers, while maintaining the improved dependability levels allowed by server replication. Two characteristics of replicated systems can be improved in order to optimize their performance: failure detection and its relation to failover triggers. The failure detection and failover settings are investigated in order to enhance the standard fault tolerance mechanisms implemented by RSerPool in a replicated IMS system.
A critical metric is the *service access time* (SAT), i.e. the average time to successfully complete a SIP transaction. By finding the right compromise between fast and accurate failure detection and tuning the failover triggers correspondingly, SAT can be improved without affecting the high dependability levels achieved with standard settings. The different parameters that should be taken into consideration during the optimization process are the following:

- Heartbeat frequency,
- Timeout of the heartbeat and SIP requests,
- Number of failovers and SIP request retransmissions;

Another important requirement for dependable solutions concerns the costs involved in the enhancements brought to the original setting. The goal is to keep the costs as low as possible. Examples of costs include:

- Overhead: additional communications due to the heartbeat mechanism impact the network capacity and the nodes' network interface and processing load;
- Implementation complexity: computational/memory resources are scarce in most handheld mobile devices and communication networks, especially in mobile networks with potentially thousands of users attached simultaneously.

The parameters of replication-based fault tolerance mechanisms are qualitatively analyzed in terms of their impact on the tradeoff between dependability, performance (IMS service access time) and additional costs (overall IMS load). Also, both the standard and replicated IMS architectures are modeled with Stochastic Activity Networks in order to evaluate the tradeoff and compare different fault tolerance parameters configurations. This way, an optimal fault tolerance configuration can be applied to a given system with specific requirements. Optimal configuration selection strategies (at

design time and at run time) are discussed and a selection example illustrates how to use the simulation results to do so.

State Consistency

Many services in communication networks are stateful, requiring that some functions collect and maintain states and/or logs about the ongoing communications, network operations and the system. In the IMS, the SIP servers are stateful entities that keep track of several states (e.g. session state, charging state). The state information is sometimes mandatory to process specific incoming requests at a SIP server (c.f. prepaid subscription example in Chapter 5). In the replicated IMS system, a backup S-CSCF takes over when the active replica crashes and in order to successfully process the next incoming requests, the selected backup server must have received a copy of the latest state information generated by the primary server before it crashed. This is called state consistency.

In distributed systems such as RSerPool, communication delays and packet losses may lead to state inconsistency, i.e. a backup server receives a SIP request before it has got the latest state update message that the primary server sent to its peer after completing the previous transaction. Many solutions have been designed to avoid inconsistency (Section 5.1) and therefore maintain the same dependability levels as with stateless replicated servers. The main drawback is that these solutions introduce latency and communication overhead that impact the service performance so there is a continuous tradeoff between consistency and service access time [Yu00]. In order to minimize the performance degradation, dynamic commitment protocols were created that apply the smallest delays necessary to reach a target consistency level in a given system [Bozinovski04a]. These protocols need to be constantly updated with inconsistency levels to dynamically tailor the state commitment delay.

An inconsistency definition tailored for the replicated IMS scenario is first given. Then, a novel inconsistency evaluation framework is proposed, which:

- permits to estimate inconsistency as previously defined and can also be used for inconsistency evaluation with a wide range of replicated entities;
- is based on contributing factors that are calculated either from the traffic and failure models of the system or using direct inputs from network analyzers. This means that the framework can be used at design time or at system run time.

The inconsistency evaluation framework is verified by comparing (1) inconsistency values measured in an experimental SIP system and (2) inconsistency values generated with the framework, where the impacting factors are calculated from a mix of inputs from the system specifications and the experiment.

The framework can also be used to help the architecture/protocol design of a system by determining the expected inconsistency levels. Accordingly, the state dissemination protocol that helps meet the predefined target inconsistency/performance tradeoff can be chosen.

1.4.2. Part II - Mid-Session Macro Handover

Macro mobility is expected to enhance the user's experience in several ways. A handover from one access network to another (so-called macro mobility, or vertical handover if the two ANs rely on different access technologies) can be triggered in the following cases:

- A better suited access network is accessible from the user's current location—e.g. to a cheaper or more energy-efficient access technology, or when more/better services are offered by a certain operator,
- Prolonged service degradation (QoS degradation or loss of connectivity) at the current access network.

In the second example, macro mobility acts as a fault tolerance solution. If failures occur on a link or at a node in an access network that does not implement systematic path redundancy or node replication, communications to and from this access network are greatly impacted, if not interrupted fully. A macro handover from the current access network (AN_{old}) to another network (AN_{new}) can ensure connectivity and potentially the desired QoS necessary for maintaining ongoing sessions. In this thesis, it is assumed that the QoS levels guaranteed at AN_{new} are at least as good as in AN_{old} before it crashed.

Terminal mobility should not be considered independently from other types of mobility because providing IP connectivity at AN_{new} is not the only requirement for a complete IMS mid-session macro handover. Indeed, on top of providing connectivity to another AN, ongoing sessions should be moved along with their corresponding states in order to set the communications in AN_{new} with similar settings as in AN_{old}—for instance the current status of an ongoing online game. This way, the session can be 'restarted' in AN_{new} where it was left pending in AN_{old} and with similar characteristics. This is referred to as session mobility. A prerequisite to session mobility is session continuity: the control mechanisms should support the changes induced by mobility, i.e. they should not prevent the session to continue despite these changes.

In the IMS, the whole session mobility could be supported by performing the standard SIP session establishment procedures at AN_{new} after obtaining connectivity. This option unfortunately raises two problems:

- According to the IMS specifications, any session should be stopped if the IP address of one of the endpoints changes during the session. This requirement causes the loss of the session states, which is unacceptable to stateful services.
- The standard IMS procedures for session establishment take time. Even though minor modifications of the standard IMS implementation would allow for changes of IP address during ongoing sessions—i.e. session continuity—the disruption time due to session re-establishment is too long not to be perceived by users.

Consequently, the standard IMS cannot provide seamless mid-session macro handover and needs enhancements to meet the users' expectations.

IMS-MIP Interworking

Mobile IP (MIP) [Johnson04] is a mature technology expected to be a standard mobility solution in IPv6 networks (cf. Section 2 in [Johnson04]), and it is shown to be faster than SIP-based mobility in many systems and scenarios, like in [Kwon02]. MIP is a network layer mobility solution whose main characteristic is to hide the handovers to the layers above it (SIP and application included) by always showing the same IP address to these layers, which makes MIP an excellent candidate to provide session continuity during mid-session inter-domain handovers. Therefore, MIP is chosen as the basis for macro mobility support in the IMS, and the SIP responsibilities are limited to the IMS session management and end-user services (e.g. SIP-based instant messaging). Unfortunately, MIP cannot be deployed in the standard IMS due to interoperability issues, so the

necessary protocol and function adjustments are proposed to make MIP and the IMS interwork smoothly and, ultimately, support session continuity. The implementation costs of the suggested adjustments are discussed. It is also shown that applying the standard MIP setup into the IMS does not reduce the handover time because MIP only implements terminal mobility and the SIP procedures still has to be completed for session mobility support from AN_{old} to AN_{new}.

Optimized MIP-Based Handover Time

The most common performance metric for mobility solutions is the service disruption time perceived by the user during the handover phase. Because all the standard SIP operations have to be successfully completed in AN_{new} before the UE is granted data bearers again, the SIP-based handover is much longer than simple terminal mobility with no session layer requirements. Integrating the standard MIP operations in the IMS provides session continuity but does not minimize the session disruption time. Hence, an enhanced MIP-based macro handover approach that supports session continuity and session mobility, and that shortens the standard, purely SIP-based, session mobility, is proposed. The solution relies on context transfer so that some secure media-authorization information obtained from AN_{old} can be reused at AN_{new}. This allows the UE to temporarily activate a unique signaling/data bearer in AN_{new} without involving SIP, which in turn considerably reduces the disruption time perceived by the user. This novel solution is described in detail and the improvements that it achieves, the implementation efforts that it implies, and its impact on the standards, are discussed. Finally, the session disruption times for the standard SIP mobility and the optimized MIP mobility are evaluated analytically and compared for a range of network delays and traffic scenarios.

Most of the contributions in this thesis have been published (cf. the *Author's Publications* section at the end of this report). To see the list of the

1.5. Thesis Outline

The thesis is organized as follows:

Chapter 2 gives an overview of the IMS, and introduces the SIP protocol and its original signaling mechanisms for multimedia session management, the interfaces between the IMS and the UMTS access network, and the IMS-specific signaling mechanisms for multimedia session management and resource allocation in association with UMTS data bearers.

After presenting the state-of-the-art on the IMS, the thesis is divided into two parts.

PART I_____IMS SERVER REPLICATION_____ _____

Chapter 3 presents the background about dependability; follows an overview of the two replication paradigms, and the functions and operations of the RSerPool and RTP replication platforms are described. A practical strategy for the integration of the RSerPool and RTP architectures in the IMS is suggested.

Chapter 4 starts with some background about the Stochastic Activity Network (SAN) formalism and the Möbius simulation tool. The standard and replicated IMS simulation models are defined as well as a set of output metrics. Several environment and fault tolerance configuration parameters are qualitatively discussed and also analyzed based on the simulation results. Methods for selecting the optimal fault tolerance configuration conclude the chapter.

Chapter 5 focuses on IMS state consistency. First, an accurate inconsistency definition for replicated IMS services is motivated, and a novel inconsistency evaluation framework is proposed and experimentally validated. Finally, the application of the evaluation framework to network design and network settings is discussed.

PART II MID-SESSION MACRO HANDOVER

Chapter 6 presents the background on macro mobility solutions, which can be used to support fault tolerance for single points of failures in the AN. The requirements for mid-session macro mobility in IMS environments are discussed. Then, the main interworking issues between the IMS and MIP are presented in detail, and a solution to overcome these issues is introduced and qualitatively analyzed.

Chapter 7 introduces a novel, optimized, MIP-based solution that shortens the inter-domain handover time in most network delay scenarios, which allows for faster recovery in case of non-replicated subsystem failures. The details about the operations that permit to reduce the handover time are given and qualitatively discussed. Finally, the new optimized MIP-based handover solution is analytically evaluated and compared to the standard SIP-based solution in order to measure the gains.

Chapter 8 draws the conclusion of this work and shows directions for future work.

2. IMS Background

IMS service provisioning is the common ground to the whole thesis. Hence, the background on IMS platforms ought to be presented—here, for the deployment scenario of UMTS access networks. First, the protocol IMS operations rely on, SIP, is presented. Then the IMS entities and architecture are introduced before the standard procedures defined by 3GPP for providing multimedia sessions to mobile users are explained in details. The standard IMS mechanisms for fault tolerance and macro mobility are summarized in Chapter 3 and Chapter 6 respectively.

2.1. IMS Paradigms

It is critical for mobile operators to control the access to their scarce radio resources. This is done by pushing towards the deployment of access and session control in order to block unauthorized users, prevent Theft of Services (ToS), and map users' subscription profiles to authorized QoS levels and the corresponding charging policies. The IMS was originally introduced in Release 5 of the UMTS specifications and provides an overlay architecture on top of access networks to support and control IP-based multimedia sessions. The IMS relies on the Session Initiation Protocol (SIP) [Rosenberg02] at the session layer to establish, modify and terminate IP multimedia sessions and to participate in the E2E session resource reservation. Therefore, the IMS is regarded as an enabler for integrating voice, multimedia, and Internet services in mobile systems.

The definition of the IMS follows three main axes:

1. The IMS was originally defined for UMTS systems but has evolved to become access independent (from Release 6), i.e. not restricted to any type of access network and can consequently be associated to fixed and mobile networks. Despite its exclusive use of IP, the IMS can nonetheless interwork with circuit-switched external networks such as 2G mobile networks and telco systems like PSTN and Integrated Services Digital Network (ISDN) via gateway functions, namely the media gateway (MGW) and media gateway control function (MGCF).

2. The IMS provides several service enablers that can be commonly used by all applications:

- Authentication and authorization
- Naming and addressing
- Control of QoS and charging
- Session management

A driver for operators to adopt the IMS is the increased probability of successful communications and guaranteed QoS via the preliminary negotiation of the capabilities supported by the networks and the end-user devices involved in the session.3GPP has put significant efforts into standardizing means for guarantying the required QoS levels for the session signaling and data traffics. The IMS provides authorization, reservation and final approval of QoS resources in the access network, and the interfaces with the functions for guarantying QoS in external IP networks (from Release 6). Service-based charging is also included in the IMS specifications.

3. The IMS provides the means for application deployment based on open standards and application programming interfaces (API). Therefore, UEs can now access applications deployed in their operator domain, in a visited domain (when roaming) or from an independent third-party application platform. The interactions between ANs and independently owned and managed application platforms assume (1) preliminary commercial agreements between operators and service providers, and (2) the mapping of these agreements into control policies (e.g. for QoS and charging). Examples of IMS services include VoIP, multi-party conferencing, audio and video streaming, push-to-talk, push-to-show, instant messaging and person-to-person gaming [O'Reagan04][Kim03].

2.2. Original IETF Session Initiation Protocol

Originally designed by the Internet Engineering Task Force (IETF) for multi-party phone conferences over the IP-based networks, SIP has been reused and extended (mainly, the INVITE session setup procedure, cf. Section 2.3.3) by 3GPP for service and session control in IMS-controlled access systems. In this section, the main characteristics of the original IETF SIP are presented: session management protocol stack, entities, messages and mechanisms. The IMS-specific SIP procedures are described in the next section.

2.2.1. SIP Overview

The Session Initiation Protocol (SIP) was defined by the IETF in RFC3261: "SIP is an application-layer control protocol that can establish, modify and terminate multimedia sessions or calls" [Rosenberg02]. SIP can be used to initiate sessions as well as invite members to sessions that have already been established so it can manage multi-party conferences.
SIP inherits features from two protocols: (1) the text-encoding scheme and header style (To, From, Date, Subject, etc.) from Simple Mail Transfer Protocol (SMTP), which is used for email and (2) the client-server design and use of Uniform Resource Locators (URLs) from Hyper-Text Transfer Protocol (HTTP), which is used for web browsing. In SIP, URLs are referred to as Uniform Resource Identifiers (URI).
Most of SIP is about the session initiation phase: initiating a session requires to determine where the user to be contacted is located at a particular moment. Once the user is located, SIP delivers a description of the session to be initiated. The most common protocol used to describe the session content is the Session Description Protocol (SDP) [Handley98]. In summary, SIP implements five major features to support the requirements for establishing and managing sessions:

- *user location*: localize the user equipment(s) to be added to a session;
- *user capabilities*: assess the possible media parameters for a session based on the capabilities of the user equipments involved in the communications;
- *user availability*: determine (beforehand) the willingness of the called users to accept the call;
- *call setup*: establish the session parameters at the user equipments (at both called and calling parties);
- *call handling*: manage call transfer, modification of session parameters, and termination.

2.2.2. SIP Protocol Stack

SIP messages can be carried over any transport protocol, such as UDP, TCP and SCTP. In order to manage the media of a session, SIP interacts with other protocols at the session layers, as shown in Figure 2.1. It is designed to collaborate with protocols such as RSVP for reserving network resources [Braden97], Real-time Transport Protocol (RTP) for transporting real-time data [Schulzrinne03], which is augmented by the RTP Control Protocol (RTCP) for providing feedback on QoS levels [Schulzrinne03].
SDP is tightly coupled with SIP for describing the multimedia sessions. The SDP message body is encapsulated in the SIP messages and carries the information needed for QoS negotiations (e.g. to select appropriate media codecs) and charging, and to send the RTP packets to the right location. Here is a non-exhaustive list of the information available at SDP:

- time(s) the session is active;
- information in order to receive data correctly (addresses, ports…);
- the transport protocol (UDP, RTP, H.320…);
- information about the bandwidth to be used for the session;
- the type of media (video, audio…);
- the format of the media (H.261 video, MPEG video, G.711 voice …).

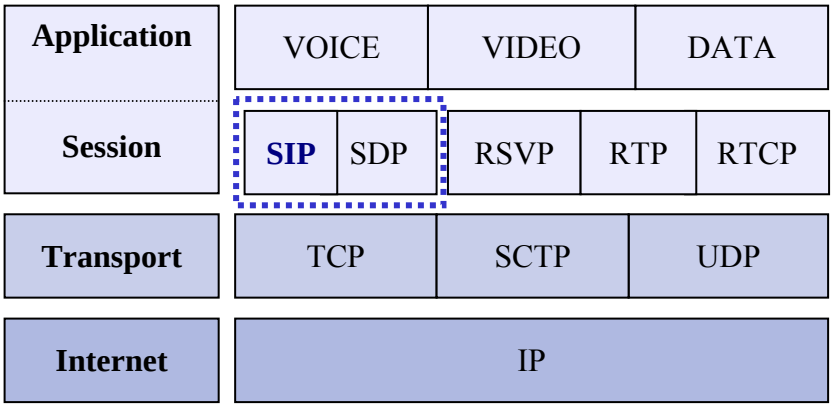


Fig. 2.1, SIP protocol stack

2.2.3. SIP Entities

The basic SIP architecture is based on the client-server paradigm. Its main entities are the user agents, the SIP gateway, and the proxy, location, redirect and registrar servers.

- A *user agent* (UA), or SIP endpoint, contains both a client function and a server function. These two functions are respectively (1) the user agent client (UAC), which initiates the SIP requests and (2) the user agent server (UAS), which generates the correct responses.
- A *SIP gateway* provides a SIP architecture with the necessary interfaces to connect to external networks utilizing different transport/signaling protocols (e.g. SS7 [ITU93] or PSTN).
- A *proxy server* receives requests, determines which server to forward them to, and then forwards them possibly after modifying some of the headers. A proxy is different from a user agent as the former does not issue requests itself.
- *Redirect servers* only respond to clients' requests and indicate—typically, to a proxy server—the next server to contact.
- The *location server* consists of a database that stores the current locations of users when they register (at the SIP level). The database can then be interrogated (e.g. by redirect servers) to provide the current address information of users.
- *Registrar Servers* are in charge of registering SIP entities by storing the registration information (entity's SIP addresses and the associated IP addresses) in a location server.

Figure 2.2 illustrates an example of SIP architecture where the SIP servers and a few user agents are deployed in the same IP network. Note that the SIP servers can be deployed in different networks. Other user agents can access the SIP services from external networks via a router (IP networks) or a gateway (fixed telco systems).

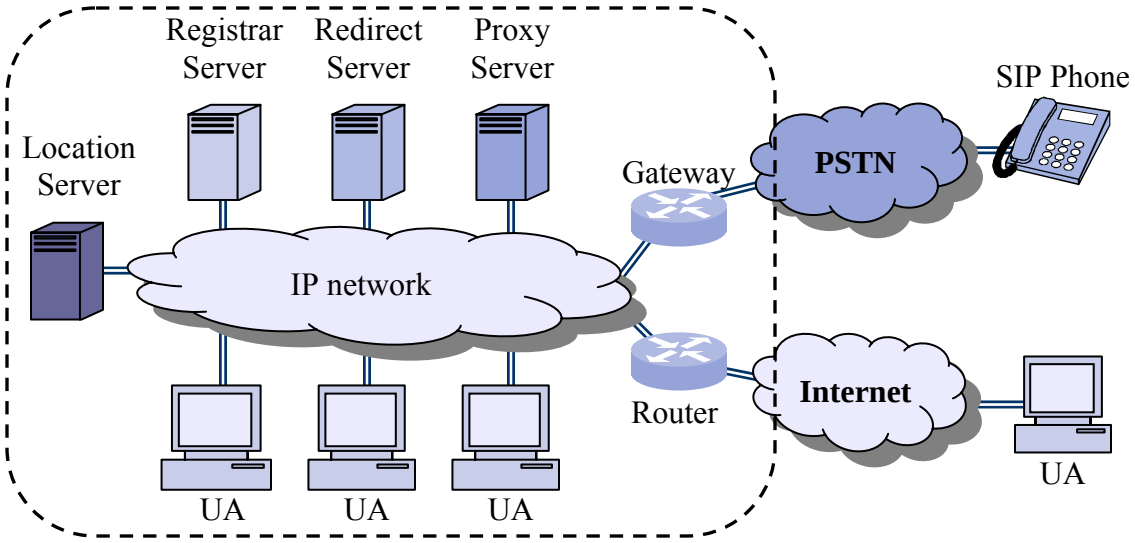


Fig. 2.2, SIP architecture

2.2.4. SIP Messages

SIP is a transactional protocol; a transaction corresponds to a request, optional provisional responses, and a final response.

SIP Requests

Among the several types of SIP requests, the most important are the REGISTER, INVITE, ACK, BYE, and CANCEL transactions:

- The *REGISTER* request is used by a user agent to notify a SIP network of its current IP address(es) and the possible URIs at which it can receive calls.
- The *INVITE* transaction is used to establish media sessions between user agents. Success responses (cf. next subsection) to INVITE requests are always acknowledged back with a final ACK request (i.e. from UAC to UAS).
- *ACK* is used to acknowledge the response to an INVITE request. It confirms that the caller has received the success response to its INVITE request and that the call can start.
- The *BYE* request is used to terminate sessions. It can only be sent any user agent participating in the session, never by proxies or other third parties.
- The *CANCEL* request cancels a pending request from another transaction but cannot affect completed transactions. A request is considered pending if the server side has not issued a final response yet; otherwise, the request is completed. CANCEL requests can be issued either by any user agent.

A SIP session is defined by the sequence of SIP transactions and application level communications within the time interval between the successful completion of an INVITE transaction and the terminating BYE transaction.

SIP Responses

A SIP response is a message generated by a UAS or a SIP server to reply to a request sent by a UAC. The different classes of SIP responses are listed in Table 2.1. A provisional response can provide information about the current status of a transaction, while a final response ends a transaction, whether it is successfully completed or not.
All SIP responses are listed in Appendix A.1.

Table 2.1, SIP response classes

Class	Description	Response	Example	Meaning
1xx	Informational	Provisional	180	Ringing
2xx	Success	Final	200	Success
3xx	Redirection	Final	302	Moved temporarily
4xx	Client error	Final	401	Unauthorized
5xx	Server error	Final	502	Bad gateway
6xx	Global failure	Final	600	Busy everywhere

SIP Headers

There are four types of SIP headers: general headers, request headers, response headers and entity headers.

- The *general headers* are all the required headers in a SIP message. General headers can be present in requests and responses. These headers are created by user agents and cannot be modified by proxies.
- The *request headers* allow the client to give the server some additional information about the request and about the client itself.
- The *response headers* allow the server side to add information about the response.
- The *entity headers* provide information about the message body.

All the SIP headers and their definitions are available in Appendix A.2.

2.2.5. SIP Mechanisms

Addressing

UAs are reached with their SIP addresses, which are identified by SIP URIs in the *To*, *From*, and *Contact* headers.

A SIP URI has the following format '*sip:user@host'*, where the user part is a username or a telephone number and the host part is either a domain name or a numerical network address. In many cases, users' SIP URIs can be guessed from their email address.

Routing and Locating SIP Entities

SIP messages are routed from one SIP entity to the next on the E2E path between UAC and UAS. This means that each SIP entity attaches a new IP header before forwarding the SIP message to the next intermediate SIP hop.

When a UAC sends a request to the SIP URI of a UAS, the first SIP node contacted is usually a pre-configured proxy in UAC's domain. This proxy is in charge of forwarding the request to a proxy in UAS's domain and the latter finally forwards the request to the UAS. In case a SIP node does not know about the IP address(es) currently mapped to the SIP URI of a UAS, it can interrogate a redirect/location server that will return a list of plausible networks or specific IP addresses where to find the SIP entity. The location server updates this information from static lists (e.g. for proxy servers locations) or during user registration procedures (a UA registers its current IP address(es)).

Changing a Media in an Existing Session

During a session, a UA may want to change the media transmitted or other session parameters. This is done by re-issuing an INVITE request, so-called re-INVITE. This request uses the same Call-ID as the ongoing session, but carries the new set of parameters to be used.

SIP Mobility and Fault Tolerance

SIP also provides means for fault tolerance and several types of mobility. They are addressed respectively in Chapter 3 and Chapter 6.

2.2.6. SIP Session Example

A SIP session example is shown in Figure 2.3. In the depicted scenario, the UAC initiates a SIP session and sends the INVITE request to its proxy server first. Here, it is assumed that the proxy server already knows from the SIP URI—put in the *To* header of the INVITE by the UAC— which actual IP address the INVITE request should be forwarded to; this is the case for instance when the proxy server has previously forwarded a request to this specific UAS on behalf of another UAC. If the proxy server did not know the current IP address of the UAS, it would have to request this information from a location server. When the UAC receives the 200OK response from the UAS, it immediately sends the final ACK back and the two endpoints can start:

- interacting directly at the SIP level, i.e. without contacting intermediate proxy servers;
- exchanging data packets at the application level. When data packets are exchanged, the SIP layer is not involved in the communications; other session layers such as RTP might be involved though, e.g. to provide QoS support to IP packets.

When any of the UAs wants to terminate the session, it initiates a BYE transaction.

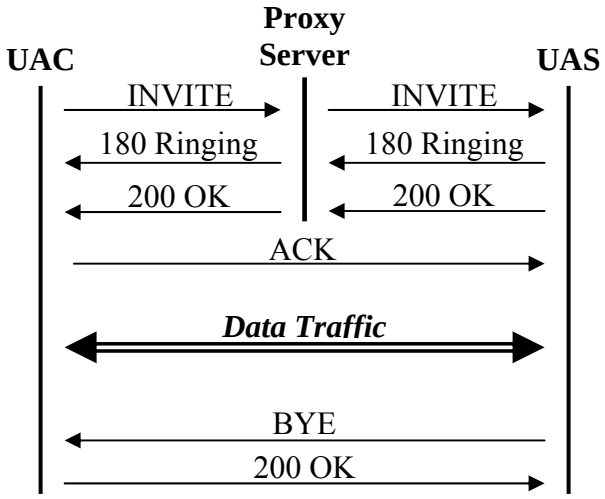


Fig. 2.3, Simple example of original IETF SIP session

2.3. IMS for 3GPP UMTS Networks

In the IMS, the signaling operations are supported by a combination of (1) SIP transactions when entities outside of the access network are involved and (2) specific internal calls for synchronizing the IMS control into the access network. Moreover, some IMS-specific SIP transactions are different than in the IETF SIP setting (c.f. Section 2.3.3), and the IMS defines a different set of signaling servers, the CSCF servers.

2.3.1. The IMS Architecture

The basic IMS architecture consists of three different types of CSCF servers [3GPP02a] plus an additional supporting database:

- *Home Subscriber Server* (HSS) is the integrated database that consists of a location server, which stores information on the location of users, and a profile database, which stores security and service profile information about subscribed users.
- *Proxy CSCF* (P-CSCF) is the server initially contacted by the SIP devices. All SIP requests are sent from the client to a P-CSCF first. The P-CSCF is usually associated to a Policy Control Function (PCF)—see next subsection—that interacts with the access router to apply operator's access control policies to each bearer in the access network. The

P-CSCF controls the access network, while being detached from the access network.

- *Interrogating CSCF* (I-CSCF) acts as first contact point from/to other IMS networks and has the additional task of selecting an appropriate S-CSCF with the help of the HSS during a user IMS registration.
- *Serving CSCF* (S-CSCF) is mainly responsible for managing user profiles and the call states. It performs service control and assists the billing functions by maintaining charging states. Furthermore, it provides interfaces to application servers.

Figure 2.4 illustrates an IMS providing control to UMTS access networks and it shows that the IMS can be deployed remotely from the UMTS ANs so long as they are connected across the Internet. Outside the UMTS, signaling flows always go via a P-CSCF before reaching the IMS. Note that after contacting the IMS for registration purposes, a UE's signaling messages can be routed directly from the P-CSCF to the S-CSCF, shortcutting the I-CSCF.

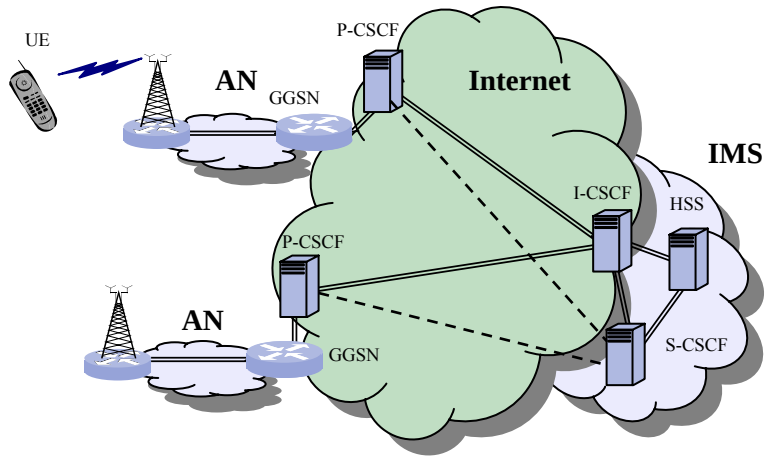


Fig. 2.4, UMTS+IMS architecture

A stateless server is an entity that does not maintain any state of the ongoing sessions or transactions when it processes SIP messages; it simply forwards each request and response it receives. A transaction stateful server is an entity that maintains states for the duration of a transaction only (e.g. when forking a request to several destinations). A call stateful or session stateful, server retains the global state of a session from the initiating INVITE transaction to the terminating BYE transaction. In common implementations, the session state is updated after the (successful or failed) completion of a transaction.
Even though this functional aspect is left open in the IMS specifications and is therefore implementation-specific, some CSCF servers may need to maintain states. This is at least mandatory at the S-CSCF, e.g. for billing and for failover purposes—in order to pick up the session at a backup S-CSCF with the same session settings and in the same state as when the session was before the active S-CSCF failure.

2.3.2. IMS Control Functions in UMTS

The UMTS specifications define all the necessary interfaces in order to apply in the access network the control policies negotiated during the IMS session setup phase. When a UE wants to send packets through a UMTS AN, it must activate a PDP context. This creates a PDP context data structure in the SGSN that the user is visiting and the GGSN serving the access point. A PDP context contains information about the session such as UE IP address, QoS parameters, Tunnel ID (i.e. routing information), etc. PDP contexts allow for QoS differentiation of IP traffic in the UMTS by using independent UMTS bearers.
For IMS services, at least two PDP contexts are created for each UE. The first PDP context allocates resources dedicated to the SIP signaling flow. The other PDP contexts are created on request for each active media stream the UE is participating to, with their respective resource reservations.
In the following, we describe the IMS functions and corresponding entities that are involved in access and session control in the access network.

Policy Control Architecture

Figure 2.5 shows the different entities that participate in access and session control.

- SGSN (Serving GPRS Support Node): The SGSN performs the necessary functions in order to handle the packet transmission to and from the UE, including mobility support within the operator domain (i.e. micro mobility).
- GGSN (Gateway GPRS Support Node): The GGSN is the network element connecting the UE to the external network. The GGSN contains a PEP to enforce policies. It also contains a UMTS BS Manager for handling resource reservation requests from the UE (e.g. through PDP context signaling).
- UMTS BS (Bearer Service) Manager: The UMTS BS Manager handles resource reservation requests from the UE during the PDP context activation procedure.
- PEP (Policy Enforcement Point): The PEP is a logical entity that enforces in the UMTS AN the policy decisions made by the PCF.
- PCF (Policy Control Function): The PCF is a logical policy decision element which implements policies in the IP media layer. The PCF makes decisions in regard to network-based IP policy using policy rules, and communicates these decisions to the PEP in the GGSN via the standard Go interface for PDP contexts setting.

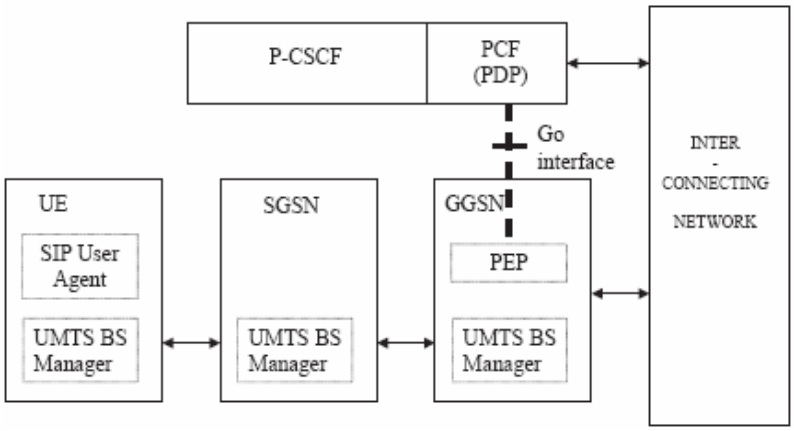


Fig. 2.5, Policy control model for UMTS

Access Control

The access to IMS services is granted after a UE successfully completes sequentially the following steps:

- GPRS Attach procedure: during this procedure (see Section 6.5 in [3GPP04b]), the UE connects to the UMTS services by authenticating itself, updating its location information in the UMTS AN and creating a security association up to the SGSN.
- PDP context activation: the PDP context creates a logical connection between the UE and the GGSN that has its specific QoS settings and is used exclusively for signaling purposes (see Section 9.2.2 in [3GPP04b]). Upon completion, the UE gets a P-CSCF IP address(es) and can then start contacting the IMS.
- SIP REGISTER transaction: the UE has to be authenticated by the IMS and does so with the REGISTER transaction (Section 5.2.2.3 in [3GPP02a]).

The UE is now able to request the setup of multimedia setup by calling the IMS/UMTS service and session control functions. Note that if a UE's profile does not allow it to be authenticated or authorized (typically when roaming in a visiting network or when the account is out of credits), no further operations are possible as no IP bearer is created.

Service Control and Session Control

The PCF is the logical entity co-located with the P-CSCF that enables general policy control over IP bearer resources and SIP services to evolve separately in the UMTS UTRAN and PS-CN. This logical policy decision element uses standard IP mechanisms to implement Service-based Local Policy (SBLP) in the bearer level. Its task is to enable the coordination between events in the SIP session level and access network resource management by authorizing QoS requests based on the user's profile. The PCF communicates with the GGSN via the Go interface to transfer information and policy decisions (following the COPS framework [Durham00]). Therefore, the GGSN is the

policy enforcement point for Service-based Local Policy control. The QoS level authorized by the PCF applies only to a specific media stream, which is defined by a unique PDP context in the UMTS. Here is some of the information that can be sent to the GGSN from the PCF:

- Destination IP address,
- Destination port number,
- Media type information,
- Bandwidth parameter.

PDP Contexts Differentiation

In UMTS, when packets are delivered over the air interface, a packet filtering function operates using a TFT (Traffic Flow Template), which is located at the GGSN and is established when configuring the radio bearer. A TFT classifies incoming packets from external networks into proper PDP contexts. In other words, the TFT filters the incoming packets received at the GGSN and selects the appropriate PDP context. The TFT hence specifies the profile of the data that should be carried by the radio bearer. A TFT can contain the following data:

- Source IP address,
- Destination port range,
- Source port range,
- IPsec Security Parameter Index (SPI)
- Traffic class
- Flow label

2.3.3. Complete Standard Service Provisioning Operations Sequence

From L2 Connectivity to IP and SIP Connectivity

Before a UE can access IP-based communications with external nodes, the legacy UMTS layer 2 (L2) mechanisms have to be performed. After gaining connectivity on the UMTS air interface, the UE triggers the GPRS Attach procedure in order to establish a logical connection in the UMTS IP core, up to the SGSN, and to set security functions.

With respect to multimedia service provisioning, the most important is the primary PDP context activation, which provides the UE with an IP address, extends the connection from the UE to a GGSN, which is in charge of managing the access to external IP networks, and allocates a dedicated P-CSCF that forwards all SIP messages from or to this UE. Ultimately, the primary PDP context is intended to act as a signaling bearer through the UMTS UTRAN and PS-CN, i.e. the primary PDP context is used for SIP/IMS signaling only and additional PDP contexts should be created for each media flow (see next subsection). Thus, the primary PDP context activation request includes information about the desired level of QoS that the UE-GGSN tunnel should guarantee for signaling traffic. If allowed by the operator, a general PDP context can be created instead that carries both signaling and data traffic.

In terms of UMTS security, UE and access network are mutually authenticated during the GPRS Attach procedure with the Authentication Key Agreement (AKA) procedure [3GPP00] and security associations are created to secure the path between UE and SGSN.

After the UE has obtained IP connectivity, it should register at the SIP level (REGISTER transaction) to be IMS-authenticated, to set security associations with the IMS, and to update its location information at the HSS (e.g. for future incoming calls). Both UMTS AKA and IMS AKA authentication procedures are explained in detail in [Zhang06]. Note that the P-CSCF maintains a security list that defines the range of IP addresses authorized for the REGISTER.

Multimedia Session Setup

The actual trigger for a multimedia session setup is the SIP INVITE transaction. Figure 2.6 shows in detail the whole INVITE message flow as defined by 3GPP. During this extensive message exchange, the UE initiates a new session whose media parameters are negotiated in UE_A's and UE_B's respective AN in order to activate the so-called UMTS data bearers. During this procedure, the E2E signaling allows for E2E QoS provisioning; the IMS defines the procedures to guarantee QoS in the UMTS and provides the necessary interfaces for QoS reservation in the Internet (out of scope in this thesis).

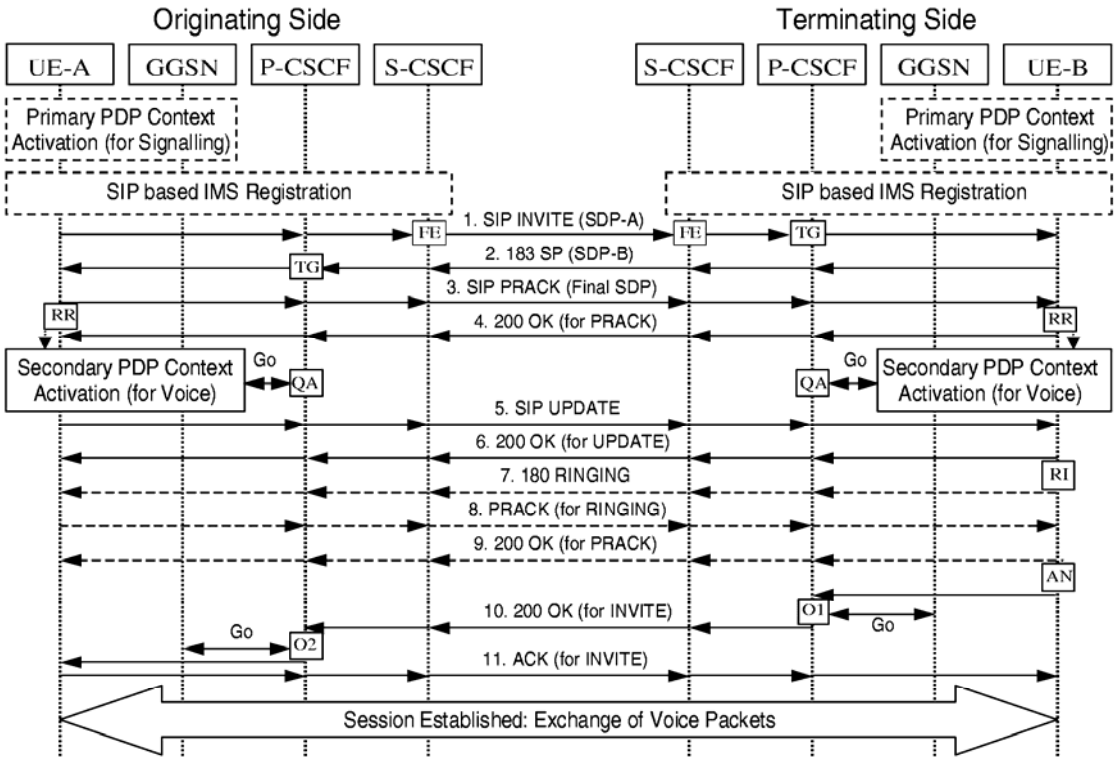


Fig. 2.6, IMS session setup flow [Kim03]

First, the two endpoints (UE_A and UE_B) negotiate the QoS level they require for the multimedia session with SDP content descriptions that are included in the body of the SIP messages (Messages 1-2). When Message 2 reaches the P-CSCF, the PCF authorizes the resources requested by the users, based on the user's subscription profile, and adds a Media Authorization Token (AuthToken) in the 183 SIP message (cf. 'TG' in the figure).

AuthToken is used to identify the PCF that authorized the resources requested by the UE. As long as the AuthToken lifetime has not expired, the UE can reuse this token in subsequent secondary PDP context activation requests it addresses to its GGSN. Based on this information, the GGSN retrieves the PCF identity and uses COPS over the Go interface to authorize the required resources from the PCF. Then, the PCF makes sure that the resources requested were authorized for this particular UE. After receiving confirmation from the PCF, the GGSN allocates the resources in the access network by activating one secondary PDP context for each media stream. The next messages (Messages 5-11) are exchanged to set the filtering functions implemented at the GGSN, namely the TFT and SBLP filters. TFT dispatches incoming packets to their respective PDP context based on the source address and SBLP is used to block outgoing and incoming packets whose destination addresses were not included in the PDP context negotiation.

PART I

—

Server Replication

3. Fault Tolerance – State-of-the-Art

As motivated in Chapter 1, this work focuses on IMS services and the necessary means to make them more dependable. In this chapter, the background on fault tolerance techniques for communication networks is presented, with the emphasis on redundancy, namely server replication, as the main solution to handle node crash faults and link faults. 'Replicating' and retransmitting requests is another technique that can prove effective against short-term faults (e.g. bursty server or network overload). The two main architectures to manage server replication, namely distributed- and cluster-based replication, are thoroughly described. Finally, an integration/mapping scenario of these two solutions into the IMS is proposed.

3.1. Fault Tolerance Schemes Overview

Fault tolerance is the ability to avoid service failures in the presence of faults. This ability usually relies on error/failure detection and system recovery, which can both be provided at several layers of the protocol stack. Some protocols of the protocol stack implement their own standard fault tolerance solutions; others do not offer any means to hide failures to the end-user: e.g. at the transport layer, UDP only provides transport functions while TCP additionally supports congestion control.
Here is a summary—layer by layer—of the fault tolerance solutions implemented at the most commonly used communication protocols in the TCP/IP model.

3.1.1. Fault Tolerance at Layer 2

Fault tolerance at layer 2 is implemented by link technologies and aims at detecting and correcting erroneous bits in the frames sent over a link. The frames are checked and potentially corrected at every single-hop. Note that these techniques are especially relevant for wireless links, since wireless channels are particularly prone to bit errors/corruption because of noise/interferences, and maintain data integrity to the upper layer functionalities.
The most common bit error detection mechanisms rely on a piece of redundant information to validate the information integrity and examples include parity bit check and Cyclic Redundancy Check (CRC) [Peterson71].
To cope with corrupted information in a frame, the receiving side may have two options:

- Forward Error Correction (FEC) employs error correcting codes at the sender side so that the receiver can correct bit errors. One way of doing FEC is to add redundant parity bits at the sender side, and these parity bits are then used by the receiver to detect and correct errors. The extent to which a FEC scheme impacts the system performance highly depends on the amount of redundant bits. The more redundant bits appended to the original data, the more error bits in a single packet can be

corrected. Even though this technique induces overhead, the fact that fewer frames are corrupted also means that fewer retransmissions occur (see next point). In most scenarios, this translates into better overall goodput than when FEC is not implemented.

- Automatic Repeat reQuest (ARQ) is used to detect bit errors and trigger Layer 2 retransmissions (i.e. only over the link where the error was introduced)—the trigger can be a timeout or a request sent by the frame receiver. Compared to FEC, ARQ is simple and achieves reasonable throughput when the channel error rate is not very high, i.e. when few retransmissions are triggered. However, ARQ quite often leads to longer E2E transmission delays due to the additional time of retransmissions. Motivated by this observation, Hybrid ARQ (HARQ)—the combination of FEC and ARQ—was developed, where ARQ is only used when an error has been detected in the received frame that cannot be corrected by an error correction scheme at the receiving side. More advanced implementations of HARQ can also reuse erroneous packets to rebuild the correct info when receiving retransmitted packets.

3.1.2. Fault Tolerance at Layer 3

As opposed to the fault tolerance techniques at Layer 2, fault tolerance at Layer 3 mainly deals with relatively long-term errors due to link breakage caused by (physical) link failures and network topology changes.

Depending on the type of routing protocol, different fault tolerance techniques should be considered. For unicast communications, fast rerouting (see e.g. [Shand07]) and multi-path routing are two example techniques. Fast rerouting addresses the problem of finding an alternative route to a broken route; in multi-path routing, multiple routes are deployed and the traffic can be selectively sent over a specific route in order to increase dependability when other routes are highly prone to errors. For broadcasting, the problem is about achieving the right balance between reliability and efficiency. Examples of dependable broadcasting techniques are analyzed in [Liu07].

Layer 3 can also make use of multi-homing, which makes a node reachable via different IP addresses, possibly obtained from different networks. This allows for even more route diversity and independence to network failures as compared to the multi-path routing scheme. L3 multi-homing is implemented by several standardized protocol and protocol extensions: Multihomed Mobile IP (M-MIP) [Åhlund03], Hash Based Addresses (HBA) [Bugnalo07], and Host Identity Protocol (HIP) [Moskovitz07].

3.1.3. Fault Tolerance at Layer 4

While Layer 2 starts retransmissions when bit errors are detected on a hop-by-hop basis, reliable transport protocols trigger retransmissions when E2E timing failures occur. At Layer 4, retransmissions are handled by connection-oriented protocols like Transmission Control Protocol (TCP) and Stream Control Transmission Protocol (SCTP) [Stewart00]. To do so, the transport protocol at the sending side activates a timeout when a message is sent. If an acknowledgment to this message has not been received before the timeout expires, the message is suspected to have not reached the receiving side and is resent. Retransmission mechanisms for the different reliable transport layer protocols usually follow the same pattern: a default value for the first retransmission timer (T_0) is set based

on the round trip time between the two endpoints and the timer value exponentially increases after each retransmission. The retransmissions stop (and the message is definitely discarded) when the timer reaches $64.T_0$ (i.e. seven unsuccessful transmissions in total).
While common timeout techniques implemented at Layer 4 usually address timing failures due to network congestion and packet losses, SCTP brings a new perspective by addressing network failures thanks to its heartbeating (for failure detection) and multi-homing (for failure recovery) features. Each SCTP endpoint—when it implements multiple network interfaces—uses a primary IP address for communications and a secondary IP address as a backup in case of timing failure. This solution provides tolerance to physical network interface failures and also provides access diversity if the network interfaces are connected to different networks.

3.1.4. Fault Tolerance at Layer 5 and Layer 7

Layer 5 provides means to negotiate and control communication aspects such as the application used (i.e. media type description), the codecs required, etc. SIP is currently the most popular session management protocol in the IP world and, like the reliable transport protocols, it handles E2E timing failures with the same exponential backoff mechanism on a request basis. The SIP timer is an estimate of the round trip time and its default value is 500ms but it is recommended it be larger in case of high latency access links. The request retransmissions cease upon reception of the appropriate response, or after a maximum of seven transmissions of the request. When starting the next transaction, the SIP resets the timer to the default time interval.
At Layer 7, a wide range of applications can be deployed. Each application is specific and can implement its own fault tolerance solutions. Layer 7 fault tolerance often uses time-based failure detection and data retransmissions (e.g. file transfer [Postel85] or e-mail application [Postel82]). Additionally, application crashes are potentially detected by the operating system at the node where the faulty application is running, which is then restarted [van Moorsel06]. The restart operation is called rollback when the application is brought back to the same state as before the restart, or a roll-forward when the old application state is lost and the application runs with a new state. In some other failure scenarios (e.g. memory leak due to erroneous service/application programming), the operating system might restart the whole system.

3.1.5. Motivation for Server Replication in the IMS,

It is crucial to minimize SIP and IMS node failures because they impact the user experience. When initiating a phone call or changing some parameters of an ongoing session, SIP mechanisms are invoked; if these procedures cannot be completed, a phone call cannot be initiated or the parameters of an ongoing session cannot be changed, even though the application (running on top of SIP) is operational.
All the standard fault tolerance solutions introduced so far mainly deal with communication aspects and re-attempt to reach the next hop (L2) or network (L3), or the other endpoint (L4, L5, L7) when the respective layer suspects that the communications are not performing as expected. This approach, despite being beneficial to the system dependability, covers only partially the fault spectrum. E.g. when an endpoint undergoes

a halt failure, the seven SIP request transmissions will not help as the endpoint is not likely to be brought back up during the overall time interval covered by the seven timeouts. In this case, multiplying the routes to the failed entity is not the answer either. To offer timely service provisioning to the users, backup servers should be available to take over the tasks of a failed server(s), whether the service is interrupted because of a complete server failure (the machine does not respond) or a partial failure (the machine can communicate, e.g. up to the SIP layer but the application is deadlocked). The usage of backup resources is referred to as redundancy, or replication.
Consequently, replication frameworks are investigated in this work in order to mask IMS server failures. In the following sections, the background on server replication techniques is given, followed by a discussion on the model for integrating the IMS and replication platforms.

3.2. Server Replication Paradigms

Redundancy deployment has been one of the most commonly used techniques to provide dependability in many industrial areas such as control of production lines and automotive industry. In communications networks, crucial system components can be 'replicated' for dependability and load-balancing purposes. Those components are called peer replicas. Typically, multiple server nodes (or processes) that implement the same service are deployed so that big computational tasks can be broken down into smaller tasks, each allocated to a replicated server, and when one instance of the processes/servers crashes, its load can be switched over to another, or several other, peer replica(s) in the given system. All the replicated servers form together what is called a server set, or server pool.
There exist two main paradigms to implement server replication in communication networks, namely the cluster approach and the distributed server approach. These two solutions are mainly aimed at increasing dependability, but they can alternatively be used to increase the capacity of a system by increasing the available computational resources (which in turns also favours availability).
Deploying redundant servers in so-called clusters has been widely applied for dependable service provisioning. Clusters provide a *single image system* to the clients, i.e. clients see the cluster as a single server and they are not aware of the internal structure of the cluster. The cluster paradigm is implemented in the middleware—between the transport layer and the application layer in the protocol stack—and the server set is traditionally deployed in the same subnetwork (i.e. in LANs). The Resilient Telco Platform (RTP) [FSC03] is an example of a recent commercial cluster solution. In the past few years, an alternative approach has emerged that relies on distributing the redundancy over different networks. This is not a requirement though and peer servers can be connected to the same network. As opposed to the cluster case, distributed server replication moves part of the failure-detection and failover functionalities into the client. IETF standardized the distributed Reliable Server Pooling (RSerPool) protocol suite at Layer 4/Layer 5 [Lei07].
Both the RSerPool and RTP schemes are introduced in detail in Sections 3.2.2 and 3.2.3 respectively.

3.2.1. Requirements for Redundant Systems

Providing dependability requires mechanisms that theoretically hide the impact of all system faults to the end user. Therefore, the main challenge is to achieve a seamless transition from the faulty state of a system to its correct state. After a fault occurs at any component of the system, an operational component should transparently take over the functionality of the failed one and the fault tolerant mechanisms should be executed as transparently as possible from the end user's perspective. Note that this requirement holds even when the fault(s) occurs during ongoing communications/sessions, which makes the requirement even more challenging. To be fault tolerant, a redundant system should implement the following crucial services—the first two functions are only required for stateful servers, i.e. servers that manage states:

- **State-sharing algorithms (SSA)**: the active server regularly replicates the states related to ongoing communications to the set of peer servers deployed in the system. The SSA service consists in managing and maintaining the states among the server pool so that an incoming state read request returns 'correct' state information according to the requirements imposed by the service/application that requested the state read. Therefore, SSA should encompass state commitment (i.e. correctly update the state) and state access (i.e. read the correct state values) mechanisms.
- **Dissemination protocol (DP)**: the dissemination protocol is the transfer mechanism of the state-sharing algorithms and its task is to communicate and distribute state updates to all peers that belong to the server set (c.f. [Bozinovski02] for a comparison of dissemination protocols for replicated IMS servers).
- **Fault-detection mechanism (FDM)**: fault detection is essential for efficient fault-tolerance as it is responsible for providing information about the state of the system to the recovery mechanisms. The failure detection should be as fast as possible in order to allow for prompt recovery strategies but the faster the failure detection, the less accurate. Therefore a compromise should be maintained between short faulty system state and potentially costly false alarms due to inaccurate failure detection.
- **Failover mechanism (FM)**: when a fault or failure is detected, the recovery mechanism triggers what it thinks is the appropriate recovery mechanism. The recovery mechanism supported in a replicated server set is a so-called failover. Its task is to switch the service provisioning to an active server within the server set according to the deployed server selection policy.
- **Server selection policy (SSP)**: it defines the next candidate server(s) in case of failover. SSP is similar to load-balancing in the sense that for every request, the load-balancing policy determines which process the request should be allocated to. As for load-balancing, the SSP can be round robin, weighted round robin, backup, persistent backup, least used, most used, etc. The SSP function requires access to the list of all the active servers in the server set. This list can be (1) statically configured or, (2) dynamically obtained and updated. The first option is simpler to implement, but there is a major drawback: in systems with frequent dynamic reconfigurations of the server set (registration/de-registration and node failures) the list quickly becomes obsolete and not representative of the active servers in the set, which lowers dependability.

3.2.2. Distributed Servers Paradigm, RSerPool

RSerPool Architecture

The goal of RSerPool is to provide the architecture and protocols for management and operation of server pools supporting highly available and reliable applications within the Internet, and for client access mechanisms to these server pools. An important characteristic of distributed architecture like RSerPool is that the peer servers can be deployed anywhere in IP networks, even in different subnetworks.

Figure 3.1 depicts the RSerPool architecture and its logical functions. Servers that implement the same service are called pool elements (PE) and form a pool that is identified by a unique pool handle (i.e. a pool identifier). The users of a server pool are referred to as pool users (PU). A third party entity, called name server (NS), or ENRP server, is in charge of registering/de-registering PEs, monitoring the pool by keeping track of the PEs' status, and to help the PUs know which PEs the requests can be sent to.

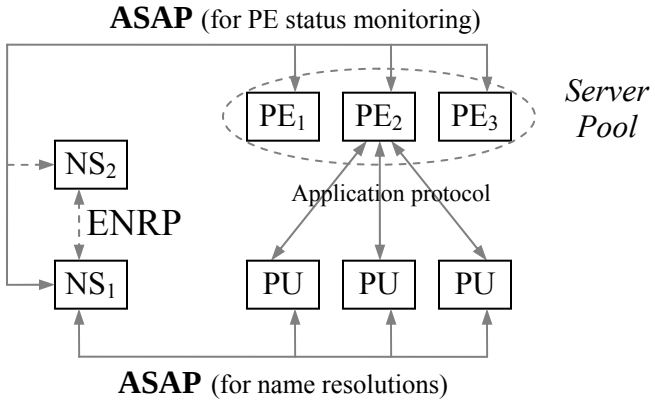


Fig. 3.1, RSerPool architecture for one pool server, where NS_1 is the default name server and NS_2 acts as a backup; NS_2 can be the main name server of another pool.

RSerPool Protocol Stacks and Functionalities Overview

Fault-tolerance in RSerPool is based on two novel protocols: Endpoint Name Resolution Protocol (ENRP) [Stewart06b] and Aggregate Server Access Protocol (ASAP) [Stewart06a]. IETF also selected SCTP to be an underlying transport layer protocol for RSerPool. This means that SCTP is used as the transport protocol for all RSerPool signaling, i.e. all messages that carry ASAP or ENRP content.

The protocol stacks for each RSerPool entity are shown in Figure 3.2. Note the two protocol stacks for the PU and PE at the transport layer: the left side shows the transport protocol for data packet transmissions—typically UDP, but it could be any transport protocol; the right side shows ASAP and SCTP, the protocols used for the RSerPool signaling packets. The name server function only implements RSerPool-related services so it does not implement several transport protocols or an application.

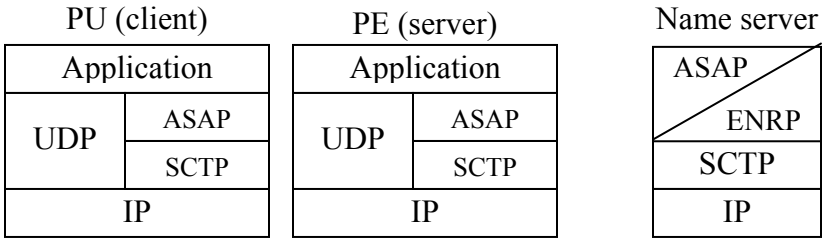


Fig. 3.2, Protocol stacks in the RSerPool architecture

PUs use ASAP to request name resolutions from the NS, i.e. the translation of a pool handle into a set of PEs' transport addresses (IP addresses and port numbers). In RSerPool, ASAP achieves similar services to DNS. As opposed to DNS, which translates a domain name in a single IP address, ASAP replies back with a set of transport addresses and a suggestion for a server selection policy. Then, the PU can keep the information obtained from the NS in a cache and use it later for choosing a server when sending future requests. ASAP is also responsible for fault-detection.
ENRP defines the procedures and message formats of a distributed, fault-resilient registry service for storing, bookkeeping, retrieving, and distributing pool membership information. Thus, ENRP communications between name servers are mainly used to disseminate the status of PEs and to share their knowledge about all server pools. Because a PE can belong to more than one pool at a time, this is needed to make sure that the information is consistent and up-to-date in every pool.

RSerPool Fault Tolerance

- **State-sharing**: [Tuexen02] requires that the name servers should not resolve a pool handle to a transport layer address of a PE that is not in operation. Thus, name servers share information about the current status of all the pools they monitor. This allows other name servers to act as backups when PUs' home name server fails and always keep the name service available. Details about how name server fault-detection and name server failovers are performed can be found in [Stewart06b].
  Note that the requirements for high availability and scalability defined in RSerPool do not imply requirements on shared state. ASAP may provide hooks to assist an application in building a mechanism to share state (e.g. a so-called cookie mechanism), but ASAP in itself will not share any state between pool elements.
- **Fault-detection**: Data loss detection is enabled in SCTP by numbering all data chunks in the sender with the so-called Transport Sequence Number (TSN). The acknowledgements sent from the receiver to the sender are based on these sequence numbers: each received SCTP packet is acknowledged by sending a Selective Acknowledgement (SACK) which reports all gaps. The SACK is contained in a specific control chunk. Whenever the sender receives four consecutive SACKs reporting the same data chunk missing, this data chunk is immediately retransmitted. Retransmissions are timer-controlled. The timer duration is derived from continuous measurements of the round trip delay. Whenever such a retransmission timer expires,

(and congestion control allows transmissions) all non-acknowledged data chunks are retransmitted and the timer is started again doubling its initial duration (like in TCP).
Another interesting feature of SCTP is the support of heartbeat messages to monitor the reachability of far-end transport addresses. An SCTP instance monitors all transmission paths to the other endpoint of the SCTP association. To this end, HEARTBEAT chunks are sent over all paths which are currently not used for the transmission of data chunks. Each HEARTBEAT chunk has to be acknowledged by a HEARTBEAT-ACK chunk. The number of events where heartbeats were not acknowledged within a certain time, or retransmission events occurred is counted on a per-association basis, and if a certain limit is exceeded (the value of which may be configurable), the peer endpoint is considered unreachable, and the association will be terminated.
ASAP has monitoring capability to test the reachability of PEs. When detecting a failure at the ASAP layer, the ASAP endpoint should report the unavailability of the specified PE by sending an ENDPOINT_UNREACHABLE message to its home NS. When the unavailability of a PE is detected at another layer, it should be reported to the ASAP layer via the Transport Failure Primitive.
Each PE is supervised by one specific name server, called the home NS. Home name servers specifically "audit" their PEs by periodically sending unicast ENDPOINT_KEEP_ALIVE messages at the ASAP layer. The NS sends this message to the PE as a “health” check. E.g., in the case when the transport level heartbeat mechanism is insufficient (usually this means that time outs are set too long or heartbeats are not frequent enough), the ASAP layer mechanism increases the probing frequency. The goal is to determine PEs’ health status in a more timely fashion. The ENDPOINT_KEEP_ALIVE_ACK message is sent by the PE to the NS as an acknowledgment to the ENDPOINT_KEEP_ALIVE message.
Using ASAP keep-alive messages also has additional value to the accuracy of SCTP fault-detection. While SCTP level heartbeats monitor the end-to-end connectivity between the two SCTP stacks, ASAP keep-alive messages monitor the end-to-end liveliness of the ASAP layer above it. This level of fault-detection implies that failures at the application layer at the PE cannot be detected, unless the application also implements a fault-detection mechanism on its own. Section XX investigates fault-detection at the application layer in the RSerPool architecture.

- **Failover**: The SCTP protocol is selected to be the underlying protocol for RSerPool due to its multi-homing capability, i.e., support of multiple IP addresses per host. Using this feature enables strong survivability in face of communication path failures, by making SCTP-enabled nodes accessible via several paths. When it detects a fault on the primary path, SCTP switches the communication over to the secondary path between the two endpoints.
  If a PE is found unreachable, ASAP can automatically select another replica in the pool and attempt to deliver the message to this particular PE. In other words, ASAP is capable of transparent failover amongst application replicas in a server pool. Practically, once the PU realizes about the failure, it can start the failover mechanism by looking in the PE list cached locally or trigger another name resolution to get an updated list from the NS. Then, the PU picks a new PE in this list and starts communicating with it. When using the cached list, the failover can be done as soon

as the fault is detected, but with some probability that an unavailable server is selected; a repeated name resolution on the other hand increases the chance to request the service to an active server but also increases the failover time.

3.2.3. Cluster Paradigm, RTP

RTP Architecture

The Resilient Telco Platform (RTP) is a middleware platform for developing dependable telecommunication applications. RTP is based on a cluster concept, which consists of several nodes which are linked together via a cluster interconnect that usually is made of simple cross-connect LAN cables. One of the primary objectives of the Resilient Telco Platform is to provide the application programmer with a single system image by using a unique external communication interface. The physical and software architectures of RTP are illustrated in Figures 3.3 and 3.4, respectively.

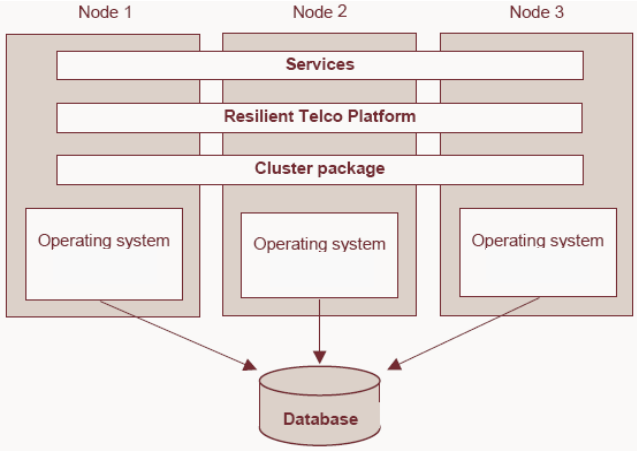


Fig. 3.3, Physical architecture of the Resilient Telco Platform [FSC03]

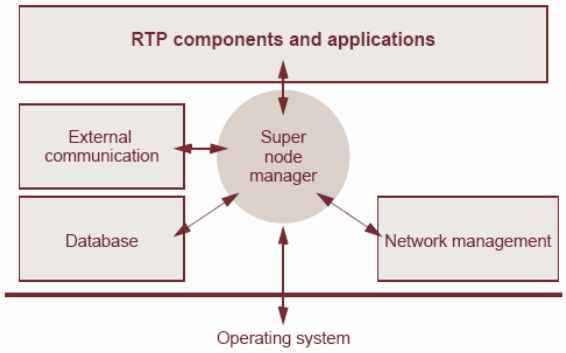


Fig. 3.4, Software architecture of the Resilient Telco Platform [FSC03]

Super Node Manager

The super node manager starts and stops the individual subsystems on the local nodes. After successful startup, it monitors them. There are two monitoring methods:

- The subsystems report themselves when they are in "serious" difficulty via "reliable" channels;
- Using health checks, the super node manager detects when a subsystem, which does not react as expected, enters an undefined state.

The super node manager then attempts to correct the error situation. This takes place in two stages:

- Local recovery, e.g. local restart of a subsystem;
- Node shutdown.

The super node manager works together with the operating system to detect the failure of another cluster node and determine the new cluster status (which nodes are currently active in the cluster and which are inactive).

RTP Components

This section describes RTP components that are of most interest for the IMS-controlled services in UMTS networks.

- **Node manager and inter-process communication**
  The *node managers* contribute to the cluster-global process management. Thus, a node manager needs to run on each cluster node. It is responsible for the management of all local RTP processes and their communication facilities. The local process management covers automatic process startup, process monitoring, automatic process restart, and process shutdown. RTP process management should not be confused with the process management of the operating system. A node manager maintains only information for processes that have *attached* to it, and the information is much different from what the operating system keeps. All node managers in a cluster need to have a common view of all active RTP processes, so that an RTP process from any node can easily address and communicate to an RTP process on any other node. Each node manager puts addressing and status information of its local RTP processes into its share of the cluster-global process table. Any change in the local process configuration is immediately distributed to all other node managers in the cluster. The validity of the global process table is periodically verified, and consistency problems are resolved.
  Although RTP processes may be spread over several cluster nodes, they can communicate to each other without having to consider the cluster architecture. The communication between RTP processes is handled by the RTP communication layer (also referred to as messaging layer or IPC layer). This layer is implemented partly in an RTP communication library as well as in the node manager. RTP processes communicate to each other via messages. Every RTP message contains a type field as a unique message identifier. When an RTP process attaches to the node manager, it is provided a queue, on which it will receive messages. The queue is established and maintained by the node manager, not by the process that owns it. If a process terminates, its queue may continue to exist. In this way, no messages are lost if the

process is immediately restarted. The address of an RTP process is simply its logical name. Since logical names are unique within the cluster, a sender does not need to distinguish between a node-local and a remote destination – the RTP communication subsystem will route the message to the receiver by the inspecting the cluster-global Process table.

- **Context manager**
  A context is data stored in memory. Contexts may be used to store "consistency points" in order to resume the software after a failure. A consistency point is e.g. the current status of a job (typically, a process of the application running over RTP).
  The Resilient Telco Platform works with a process pool that enables a large number of jobs to be processed in parallel. Before a process can begin a new job or dialog, the context of the old job must first be saved. This makes it possible to continue processing the next dialog step of the old job at a later point in time. In our case, the context manager will copy the state of the last SIP transaction.
  For performance reasons, instead of storing the context in a database, the concept of mirroring (replication) context data on another cluster node has been chosen: one node holds the master copy of the context and another one the backup copy.
  In the general case, there is a context manager master process running on a node and a backup process running on a different node. To achieve good performance, it is recommended that any operation on a context always takes place on the node where the master copy is located. Any access to a context from any other node will result in a remote access to the master and will therefore impact the performance.
  If the node with the master context manager is no longer available, the backup context manager will take over its responsibility. If the node will be available again, the master context manager synchronizes its context data with the backup context manager and then resumes its old role.

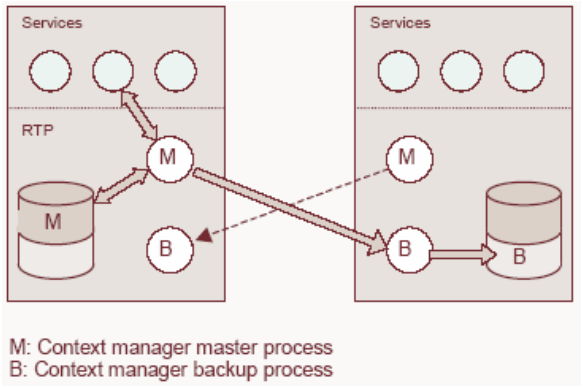


Fig.3.5, Context manager master and backup processes [FSC03]

External Communications

- **UDP dispatcher**
  The task of the UDP dispatcher is to act as a mediator between the RTP internal message system and systems outside of RTP using UDP. It will accept UDP datagrams on specified ports, analyze these messages and distribute them to RTP client applications according to algorithms implemented in the RTP UDP plugin library. Furthermore, it will accept special RTP internal messages from the RTP clients and send these messages as UDP datagrams to clients on specified ports.
  The UDP dispatcher process is started by the node manager in multiple instances. Each instance has a different logical name and command parameter that defines which protocol (MGCP, SIP, etc.) the instance handles.

  The RTP client applications must attach to a UDP dispatcher in order to use its service. Attaching/detaching means that the RTP client applications call an appropriate library function that is part of the RTP core. These library functions transparently provide the attaching service using other RTP components. The library functions also accept certain input parameters that are needed by the plugin for processing the UDP messages, which are handed over to the dispatcher.

- **Plugin library**
  The UDP dispatcher dynamically loads a plugin library, which is responsible for analyzing a raw UDP message and returning key values to the dispatcher. The dispatcher uses these values to ensure that messages of a certain type are always distributed to the same RTP client application. The name of the plugin library depends on the protocol name.
  The plugin library is not part of the RTP core and must be provided by the customer. Thus, it can be easily modified (e.g. to handle new protocols). The plugin library must provide functions for initializing, establishing, and removing special protocol key mappings, and preparing outgoing messages.

3.2.4. Integration of Replication Platforms in the IMS

Preliminary Discussions

When integrating a replication platform in the IMS, decisions should be made as to how the logical and physical components of the replication platform are mapped into the IMS architecture. Those decisions are expected to be influenced by the IMS dependability requirements and IMS traffic model.
Ideally, all IMS entities would be replicated in order to provide optimal fault-tolerance and to not introduce any single point of failure in the overall dependable architecture. Note this is generally not realistic because it is costly to implement all hardware and software components multiple times and such scope of redundancy might flood the system with pool management overhead and affect the overall performance. In this work, only the S-CSCF is replicated; this is motivated by two main reasons:

- Among all IMS processes, the S-CSCF is the most vital one since it is the logical function responsible for the negotiation and the granting of access, service and session control.
- The S-CSCF is, with the P-CSCF, the only IMS server used in every signaling flow between a UE and the IMS, and between the UE and other UE(s). P-CSCF replication is not investigated but similar conclusions as those obtained with S-CSCF replication (c.f. Chapters 4 and 5) can apply to the P-CSCF. Also, a solution is investigated in Part II to get around failures of single points in the system by deploying macro handover support.

IMS-RTP Integration

In the RTP case, integration is quite straightforward as the presence of multiple S-CSCF servers is hidden to the rest of the system. Therefore, nothing is different compared to the case without replication, except that the SIP layer in IMS entities tries to contact 'the S-CSCF' at a virtual IP address instead of the IP address of the physical S-CSCF reached. The virtual IP address uniquely represents the cluster and messages sent to this address are intercepted and parsed by the UDP dispatcher to send the message to the right S-CSCF process.

IMS-RSerPool Integration

Being the replicated entity, the S-CSCF is naturally equivalent to a PE. The NS has no direct equivalent in the IMS so it either be an additional independent server, or collocated with an existing IMS server if required by the operator. In the latter case, the HSS is probably a good candidate as it is occasionally (at SIP registration time) interrogated by other IMS entities the list of contactable S-CSCF servers. The PU, i.e. the RSerPool client, would be expected to be implemented in the IMS client as well, namely the UE. Nevertheless, one of 3GPP premises is "intelligent core network, simple terminals", to save power consumption in the terminals for instance. Thus, moving the intelligence to the access network would meet this premise. Since the P-CSCF forwards all SIP messages on behalf of the UE, the former can decide which S-CSCF in the pool the SIP messages should be sent to, and trigger retransmissions and failovers. This is the approach taken in the rest of the thesis. Note that in this setting, the dependability parameters at the UE should be tuned properly so that, e.g., there is no concurrent retransmissions of the SIP layer at the UE and the ASAP layer in the PU/P-CSCF.

4. Optimal Fault Tolerance Configuration with Replicated SIP Servers

4.1. Motivation and Problem Statement

The main goal of fault-tolerant solutions in communication networks is to increase transaction dependability. These solutions often rely on intra- and inter-node failure detection and recovery mechanisms that affect the overall performance of the systems which they are deployed in. Server replication requires that an entity frequently checks on the servers states so that service requests are most likely routed to available servers at the first attempt. This is typically done by regularly exchanging heartbeats between the servers and the monitoring entity, at the cost of traffic overhead and additional computational load in the servers. In RSerPool, the pool state information is even communicated to the clients, which is another performance impacting factor. Therefore, special attention should be given to maintaining high performance in the systems that deploy fault tolerance solutions, at design-time and—if possible—also at run-time by fine-tuning the dependability tradeoff according to the current network characteristics. In this work, dependability and performance are mainly analyzed from the user's perspective, meaning that the output metrics that are evaluated reflect on how the user will perceive the quality of the IMS services both in terms of dependability and performance.

The SIP service selected for this research consists of a single transaction and mimics SIP-based services such as registration (REGISTER transaction), SIP server capability query (OPTION transaction) or the Instant Message service (IM transaction). The latter service is similar to a chat service and the text sent by the end-users is carried in the payload of the SIP packets. For most of these services, UEs can send requests 'on-the-fly', i.e. they do not need to preliminarily initiate a session to do so. Two remarks:

- The notion of reliability cannot be applied to the service considered; once the transaction is completed, the service has been successfully provided and cannot fail anymore. This means that, in this context, dependability is equivalent to availability only.
- This type of services is very similar to other client-server services such as Internet services. Therefore, the results derived from this analysis can be reused for server replication applied to dependable Internet service deployment scenarios.

One of the main impact of node failures and network failures on user-level performance is longer transaction completion time—here referred to as service access time (SAT)—because they cause request retransmissions. Thus, SAT can be minimized if clients ideally contact an available server every time they send a request. This can be done only if the system accurately suspects which servers are down and quickly notifies the clients.

Another important performance-related aspect to consider with fault-tolerant services is the overall load in the system generated by the service platform. Even though, server replication is expected to increase dependability and reduce the number of

retransmissions, additional communications are required to support the necessary failure detection and recovery functions and it is therefore often the case that the overall load of dependable services exceeds that of the non-replicated scenario. Telco systems in general and wireless systems in particular, have limited bandwidth and perform worse with heavier loads, which in turn slows the end-user service. The goal here is not to analyze the impact of the load on the service performance, so only the overall load will be evaluated; no existing model was used—or new model derived—in order to map load levels into corresponding performance levels.
The complexity of the system modeled—including fault tolerance, traffic and failure models—is too high to compute the output metrics analytically; instead, Möbius [Möbius07] is used to model, and simulate, our system. Möbius is a tool that supports many formalisms, but was originally designed for stochastic activity networks (SAN) [Meyer85], to model communication systems and many other types of systems.

In this chapter, the theory on SAN and Möbius is introduced first. Next, the three output metrics, dependability, SAT, and load, are evaluated in the standard SIP scenario for different combinations of server and network failures. This gives a reference to subsequently evaluate the impact of server replication on the end-user experience. Then, the RSerPool-like Möbius model is explained at high-level – the detailed Möbius models and code are shown in Appendices B.1, B.2 and B.3. The fault tolerance parameters and schemes that can be tuned are presented and qualitatively discussed in order to get a first feeling about which are expected to have a bigger impact on the output metrics and should therefore be investigated more thoroughly. The most relevant results are shown and analyzed in order to draw the first conclusions on which fault tolerance settings are more appropriate to specific failure scenarios.
Because the analysis is based on three output metrics, it is not likely to find a setting for which all output metrics values are optimal. Hence, when it comes to selecting a fault tolerance setting for a real system, the best tradeoff between the output metrics should be selected. The definition of 'best tradeoff' is subjective and depends on the requirements specific to the system and end-user service. It is discussed how requirements on dependability, SAT and load allow to construct a score function that returns a unique metric, which quantifies the 'quality' of the tradeoff for each setting/scenario. A score function example is provided to illustrate how to choose one particular setting in a given deployment scenario.

4.2. Background on SAN Modeling and Möbius

As SAN models and the associated Möbius tool made the evaluation of the complex RSerPool-IMS system possible, it is crucial to understand the principles behind SAN modeling.

4.2.1. Möbius Overview

Möbius is a software tool for modeling the behavior of complex systems. The first step in the model construction process is to generate a SAN model (cf. summary in next section or more details in [Meyer85]). The most basic model in the framework is called an

atomic model, and is built with state variables and actions: state variables hold state information about a model, while actions provide the mechanism for changing model states—so-called activities in SAN.
If the model being constructed is intended to be part of a larger model, then the next step is to compose it with other models to form a larger model. This is sometimes used as a convenient technique to make the model modular and easier to construct. Although a composed model is a single model with its own state space, it is not a 'flat' model; it is hierarchically built from submodels.
After a composed model is created, the next step is to specify some measures of interest on the model using some reward specification formalism: the Möbius tool captures this pattern by having a separate model type, called reward model, which augments composed models with reward variables.
The next step is typically to create a solver to compute a solution to the reward model: a solver is any mechanism that calculates the solution to reward variables. The computed solution to a reward variable is called a result: since the reward variable is a random variable, the result is expressed as some characteristic of a random variable (this may be, for example, the mean, variance, or distribution of the reward variable).

4.2.2. Atomic SAN Models

This section contains a brief recall of the SAN primitive objects: places, activities, input gates, and output gates. These objects and their usage is illustrated in Appendix B.1.
Places represent the state of the modeled system; they are represented graphically as circles. Each place contains a certain number of tokens, which represents the marking of the place. Note that tokens in a place are homogeneous, in that only the number of tokens in a place is known; there is no identification of different kinds of tokens within a place.
Activities represent actions in the modeled system that take some specified amount of time to complete. There are two types of activities: timed and instantaneous. Timed activities have durations that impact the performance of the modeled system (such as a communication delay or the time associated with a retransmission timer); they are represented graphically as thick vertical lines. Activity time distribution functions can be generally distributed random variables, where each distribution can depend on the marking of the network. Instantaneous activities represent actions that complete immediately when enabled in the system; they are represented graphically as thin vertical lines.
Case probabilities, represented graphically as circles on the right side of an activity, model uncertainty associated with the completion of an activity; each case stands for a possible outcome (e.g. a routing choice in a network, or a failure mode in a faulty system). Each activity has a probability distribution, called the "case distribution", associated with its cases; this distribution can depend on the marking of the network at the moment of completion of an activity. If no circles are shown on an activity, one case is assumed with a probability of one. Each activity has also a reactivation function; this function gives marking dependent conditions under which an activity is reactivated. Reactivation of an activated activity means that the activity is aborted and that a new activity time is immediately obtained from the activity time distribution.
Input gates control the enabling of activities and define the marking changes that will occur when an activity completes. Input gates are represented graphically as triangles; an

arc is connected to the controlled activity, other arcs are connected to the places upon which the gate depends, also called input places. Each input gate is defined with an enabling predicate and a function. The enabling predicate is a Boolean function that controls whether the connected activity is enabled; it can be any function of the markings of the input places. The input gate function defines the marking changes that occur when the activity completes. If a place is directly connected to an activity with an arc, it is equivalent to an input gate with a predicate that enables the activity whenever the place has more than zero tokens along with a function that decrements the marking of the place whenever the activity fires.
Output gates define the marking changes that will occur when activities complete. The only difference between output gates and input gates is that the former are associated with a single case of the activity. An output gate is represented graphically as a triangle with its flat side connected to an activity (or a case of an activity); on the other side of the triangle is a set of arcs to the places affected by the marking changes. An output gate is defined only with a function: the function defines the marking changes that occur when the activity completes. There is also a default scenario for output gates; if an activity is directly connected to a place, it is equivalent to an activity in which an output gate has a function that increments the marking of the place whenever the activity is fired.

4.2.3. Composed Models

The Möbius framework allows the construction of composed models from previously defined models, which allows the modeler to adopt a hierarchical approach to modeling by constructing submodels as meaningful units and then placing them together to construct a model of a system.
Model composition is accomplished by the state-sharing approach, which links submodels together by identifying sets of state variables. Then, interactions between the submodels are possible, since both can read from and write to the identified common state variable. This form of state-sharing is known as equivalence sharing, since both submodels have the same relationship to the shared state variable.
The composed model formalism used by Möbius for SAN models is ‘Replicate/Join’: this formalism permits to define a composed model in the form of a tree, in which each leaf node is a predefined atomic or composed model, and each non-leaf node is classified as either a Join node or a Replicate node. A Join is used to compose two or more submodels using equivalence sharing; a Replicate is used to construct a model consisting of a number of identical (indistinguishable) copies of its single child—note that in the server replication scenario, a special function was created in the replicated atomic model used for the IMS servers in order to give each replica a unique ID. Each child node of a Replicate or Join node can be a Replicate, a Join, or a single atomic or composed model.

4.2.4. Reward Models

Reward models are built upon atomic and composed models, equipping them with the specification of performance measures. Möbius implements a reward model called a performance variable, which allows for the specification of a measure on one or both of the following:

- the states of the model, giving a rate reward performance variable;

- action completions, giving an impulse reward performance variable.

A rate reward is a function of the state of the system at an instant of time. An impulse reward is a function of both the state of the system and the identity of an action that completes; an impulse reward is evaluated when that particular action completes. A performance variable can be specified to be measured at an instant of time, to be accumulated over a period of time, or to be time-averaged over a period of time.
Once the rate and impulse rewards are defined, the desired statistics on the measure must be specified. Möbius includes solving for the mean, variance, distribution of the measure, or the probability that the measure will fall within a specified range.

4.2.5. Solver

Möbius supports two classes of solution techniques: discrete event simulation and state-based, analytical/numerical techniques. Any model specified using Möbius may be solved using simulation, whilst only models having delays that are all exponentially distributed, or having no more than one concurrently enabled deterministic delay, may be solved using a variety of analytic techniques applied to a generated state space.
Möbius simulation supports two modes of discrete event simulation: transient and steady-state. In the transient mode, the simulator uses the independent replication technique to obtain statistical information about the specified reward variables. In the steady-state mode, the simulator uses batch means with deletion of an initial transient period to solve for steady-state, instant-of-time variables. Estimates available during simulation include mean, variance, interval, and distributions. Confidence intervals are computed for all estimates.

4.3. IMS Server Replication - Model Definition

In this section, the different functions of the IMS replicated server scenario that were modeled and the assumptions made are defined.

4.3.1. Topology

In the scenario considered here, multiple clients want to access an IMS service from a set of replicated S-CSCF servers. The replication platform is assumed to be RSerPool, which requires the presence of an additional entity, namely the name server (NS). The NS is mainly in charge of managing the (de-)registration of the S-CSCF servers (PEs), regularly checking on the S-CSCF for failure detection, and reporting the pool status to the pool users (PUs). In order to avoid state space explosion:

- The E2E communications are between the PUs and S-CSCF servers—in the instant message application scenario, E2E communications would be between the PU and another UE, but all requests and responses would still go through the S-CSCF. Therefore, the model designed for this work can be easily adapted to model UEs as endpoints instead of the S-CSCF by simply increasing the communications delays between the PU and the other endpoint, and potentially increase the packet error/loss rate because there are more hops on the PU/UE route than there are on the PU/S-CSCF route.
- Intermediate entities between PUs and S-CSCFs, such as the I-CSCF, are abstracted.

- (and also because only the entities assumed faulty are the PEs/S-CSCF) The NS is not replicated, even though the RSerPool supports NS replication for increased dependability and more accurate pool status monitoring (cf. Section 3.2.2 on NS replication background).

The network architecture modeled with SAN/Möbius is illustrated in Figure 4.1 for m PUs and n PEs.

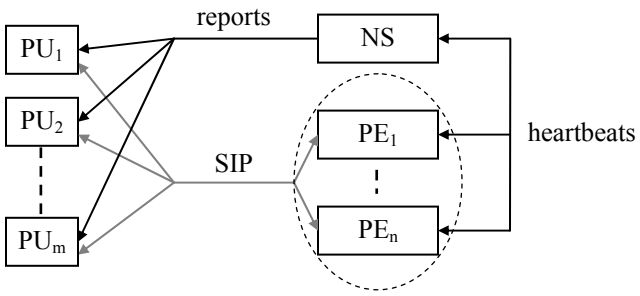


Fig. 4.1, Network topology of the RSerPool-based replicated IMS model.

The transposition of the network architecture into a SAN composed model is shown in Figure 4.2 for the example of the Regular report scheme (cf. Section 4.5.3 on report schemes) but the composed models for other report schemes would look just the same. For more details on the SAN models, see Appendices B.1, B.2 and B.3.

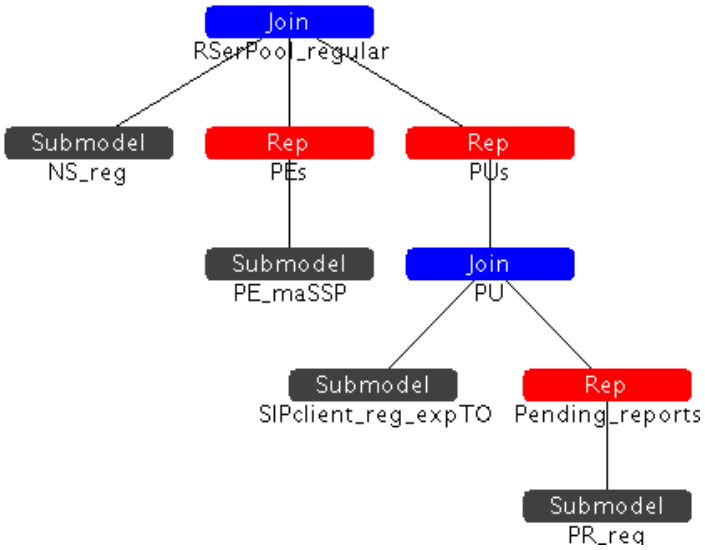


Fig. 4.2, Complete composed model for the replicated IMS

Remark on Communications Modeling Approach

When a heartbeat or a SIP request/response is generated, it is discarded before the next is sent; so at any time there are no multiple parallel pending messages between each PE and the NS (for heartbeats), and between each PU and the serving PE (for SIP transactions). Also, the link characteristics for all PE/NS and PU/PE pairs are the same – uplink and downlink are modeled by:

- Exponentially distributed communication delays, with mean value *Delay* ms,
- Packet loss probability *PER*

Because of the aforementioned traffic model, the SIP and heartbeat 'communication channels' are directly implemented by a single submodel within each replica of the PE and SIPclient atomic models.

With respect to the pool status reports, multiple messages between the NS and a given PU could co-exist simultaneously as there is no report timeout. To model this behavior, each report is allocated a specific Pending_Report replica.

4.3.2. Traffic Model

Each PU sends a new SIP request to the S-CSCF a few seconds after the last transaction initiated has been successfully completed or dropped because the maximum number of retransmissions has been reached. The inter-transaction time follows an exponential distribution, with mean value *InterSIP* seconds.

Like in the real setting, a PU can start SIP requests only after it has received a first pool status report, so-called name resolution in RSerPool. Without a report, the PU does not know which IP address(es) it should send the SIP requests to.

Note that all characteristics of the SAN model are static in the sense that they do not vary with the current state of the composed model. For instance, the current load in the system does not affect the PER or the one-way time distribution and mean. Also, processing times at the network entities were not modeled, which is not a problem because the one-way time distribution could account for processing times – from the correlation of the actual one-way time distribution and the processing time distribution at the receiver side.

4.3.3. Fault Model

The model includes three types of faults:

- Any hardware or software fault at an S-CSCF leads to the node failure (node crash faults). These faults follow an exponentially distributed ON/OFF model, with mean time to failure of *TTF* seconds and mean time to repair of *TTR* seconds. The probability that a single PE is OFF or ON is P_{OFF} and P_{ON} respectively.
- Network congestion (e.g. router buffer overflow) and bit errors on the wireless links both lead to packet losses. It is assumed that each heartbeat, report, and SIP message consists of only one packet. This is realistic because most SIP messages, such as INVITE and BYE, do not exceed a few hundred bytes (cf. Table 4.5 in [Fathi06]), and heartbeats and report messages are expected to be simpler messages than SIP requests/responses (e.g. SIP relies on many headers). Therefore, each packet loss leads to the loss of the whole message. Packet losses occur with the probability *PER* on both the uplink and the downlink.

- Network delays are exponentially distributed with mean value *Delay* for both the uplink and the downlink. Because of the delay distribution, it can happen that some heartbeat and SIP messages are delayed for a longer time than their respective timeout allows; these messages are discarded because of timing failures.

PEs are assumed stateless so no inconsistency requirement needs to be considered when defining the dependability metric. Also, it is assumed that the information returned by all entities in the system is always trusted (e.g. the content of pool status reports sent by the NS to the PUs).

4.3.4. Failure Detection and Reports

SIP provides a timeout-per-request for 'reactive' failure detection and an exponential backoff retransmission mechanism for recovery: when a PU sends a new request, it also starts the timeout-per-request, which is set to T_0 first. By default, T_0 is equal to the current mean round trip time (*RTT*), i.e. twice as much as the mean one-way time *Delay*. If the response has not been received within T_0, the PU sends the request once more but this time the timeout is set to $2*T_0$. Every time the timeout fires, the request is sent again and the timeout value doubles; the timeout value is therefore equal to $2^{(\#\ request\_retrans)}*T_0$. By default, a SIP client should send the same request up to seven times before it discards the transaction, i.e. the transaction is failed. Adding up all the successive timeout values, a transaction can live up to 127*RTT. For instance, for an estimated round trip time of 500ms, the maximum transaction lifetime is 63.5s.

On top of the reactive timeout-per-request mechanism of SIP that is implemented locally in the PUs, the NS sends ASAP-layer heartbeats to all the registered PEs every *InterHB* seconds to check on their status—the PEs are assumed to be registered when the simulation runs start. Once the heartbeat timeout expires, the NS gathers in *PElist* the identities of the PEs that have responded on time to the last heartbeat request and orders them according to their respective heartbeat response times (cf. next subsection for details on PElist ordering strategies).

In RSerPool, the recovery mechanism is implemented by the PUs since they are in charge of the request retransmissions and the failovers. Therefore, a mechanism to report the current pool status to the PUs is needed. Different reporting approaches are discussed in Section 4.5.3.

4.3.5. Failover Management and Server Selection Policy

The recovery functionalities are implemented at the PUs. First, the PUs are in charge of triggering the SIP request retransmissions when timeouts expire. After [1+*max_retrans*] unsuccessful transmissions, the PU takes the PE out of the current PElist it caches and makes a failover by sending the following request transmission to the next server in the PElist. Once *max_FO* failovers have been made and the [max_retrans+1] attempts with the last PE have failed, a transaction is discarded and considered failed. In the more likely case when the transaction succeeds, the current retrans and FO counters are reset and the same PE is used for the next transaction, unless a new report was received during the inter-transaction time and the selected PE now appears to be unavailable (i.e. the PE contact information is in the report but an OFF-flag shows that the NS suspects it to be down). Even when all PEs left in the list have OFF-

flags, the PU keeps following the retrans/FO pattern. Only in the rare case when the PElist is empty (no PE responded on time to the last heartbeat, or the PU has taken them out of the list one by one), the PU stops sending the current request and the number of failed transaction is incremented. The PU waits for InterSIP and then, either the PU has received a new non-empty report and sends a request to the first PE, or the PU has not received a report and it increments the failed transaction counter and waits for another InterSIP.
When a failover is triggered, many strategies for selecting the next PE can be chosen. Among them, relevant examples are: round robin, most recently repaired, most available, shortest RTT, etc. For replicated servers managed by RSerPool, [Bozinovski04b] has shown that the server selection policy (SSP) that offers the highest dependability levels is the so-called maximum availability SSP (maSSP): the PE that replied last to the heartbeat request is picked first, the PE whose heartbeat response was received by the NS second to last is picked next, and so on – this is a LIFO queue. Consequently, maSSP was the only server selection policy modeled and the NS orders the PElist accordingly at the end of each heartbeat round.

4.3.6. Output Metrics

Since the goal of this work is to investigate the dependability/performance tradeoff in the RSerPool+IMS context, multiple metrics are necessary.

Dependability

Dependability is simply defined as the ratio of successful transactions over the total number of transactions. This value reflects on how well a level of failure detection, associated to a specific combination of max_retrans and max_FO, permits to avoid service unavailability for a given amount of node and network failures.

SAT

SAT is defined as the average time between the moment a transaction is triggered (i.e. the moment when the request is first sent) and the moment the transaction is completed (i.e. the moment when the PU receives the response). An important remark is that SAT is measured only when transactions are successful. This is because the user's experience is mainly impacted by (1) how likely he/she will get access to the service when requested (i.e. dependability) and (2) if the service is obtained, how long it took to access the service; if the service is not available, it matters little whether the transaction is dropped after X or Y seconds.
Obviously, SAT is proportional to the RTT value—the longer the communications delays, the longer the service access times. Thus, in order to better compare how SAT varies with different RTT input values, the SAT results are normalized to RTT=100ms, i.e. $SAT_{norm}=SAT*(100/RTT)$.

Load

Even though the model does not offer dynamic parameter settings in relation to the current system load, it is important to get an idea of how much traffic specific failure detection and/or recovery strategy settings produce as compared to other settings. Hence, the load is a measure that includes all types of traffic, even failed transactions: it is

expected that failure detection decreases the SIP traffic by reducing the number of request retransmissions; at the same time, failure detection introduces a lot of overhead so it is necessary to evaluate the load in order to see if the overall traffic can be reduced when using failure detection.
Note that once a message has been sent, the load counter is incremented, even if the message is lost because of PER before reaching the receiving side. Also, each message type is defined by a specific packet size, where:

- Size_HB = 1 packet unit
- Size_report = 2 packet units
- Size_SIP message = 4 packet units

Since the $(n+1)^{th}$ transaction initiation time does not only depend on InterSIP time—it also depends on the n^{th} transaction SAT—all tests do not generate the same total number of transactions within a given simulated time. In this context, it would be 'unfair' to compare the overall load of different settings/scenarios using the total load. Instead, the overall load is normalized to one transaction and, thus, the load is given in packet units per transaction.

4.4. Input Variable Selection – Parametric Analysis

Now that the output metrics have been defined, it is important to identify the input metrics that would most impact the results. In this section, a set of input metrics for the RSerPool-based server replication is identified. These input metrics are discussed qualitatively in terms of their respective relevance in the testing strategy and how varying their values is expected to affect the three output metrics. The summary of this discussion is given in Table 4.1.

Each test run was set to last twenty hours (i.e. 72000 seconds) of RSerPool+IMS simulated operation time. All setting scenarios evaluated are run a minimum of 6 times. If some reward variables have not converged, the scenario is run again, up to 12 times in total. Note that the few reward variables that do not converge after the 12 runs are variables that only serve testing purposes, such as the number of requests that were not sent by the PUs because their respective PElist was temporarily empty. With this experiment design, each scenario tested could run up to a whole hour.

4.4.1. Influence of the System State

The influence the current characteristics of a system have on the output metrics is qualitatively discussed first. Then, results obtained for different settings of the fault tolerance solution illustrate how changes in the environment characteristics affect the output metrics.

Traffic Load Parameters

The way the system is modeled, the overall load does not impair the performance of any function modeled. Consequently, dependability and service access time are expected to be independent from the load levels in the model.

Nevertheless, the load-related input settings should be so that there are enough event samples to return statistically significant results. The overall load is made up of the heartbeat load, the report load and the SIP load:

- The heartbeat load directly depends on the heartbeat frequency. With frequency values typically ranging from 1/60 to 1/2 (cf. Section 4.5.2 for the discussion on heartbeat frequency settings), there are respectively between 1200 and 36000 heartbeats 'rounds' during every run – at each round, the NS sends a heartbeat to all the PEs in the server pool.
- The report load depends on the report scheme applied and its associated report frequency. Even though the report frequency does not have to follow that of the heartbeat mechanism, there are also between 1200 and 36000 report rounds during every run – at each round, the NS sends a report to all the PUs.
- The SIP load depends on the number of PUs (#PU) and the frequency of SIP transaction initiations (1/InterSIP). These settings were fixed to 10 PUs and 5s InterSIP time for all the runs; a total of more or less 140,000 transactions are generated during each run. The fluctuations of total number of transaction from one test setting to another are due to the SIP retransmissions, which affect SAT and the transaction initiation times distribution.

The total numbers of sample for each test scenario are actually even bigger than the ones given above because each test is run from 6 to 12 times. These numbers show that the selected load-related settings generate enough samples to confidently derive conclusions from the evaluated results.

The average load per transaction is a function of the SIP traffic load setting. This is because for each test scenario, the heartbeat and report frequencies are fixed and, thus, their contribution to the total load is fixed too. Since the SIP traffic settings influence the number of transactions generated during each run, these settings also influence how much the heartbeat and report traffics contribute to the average load per transaction.; the more SIP transactions, the lower the load per transaction. Let us verify it analytically:

$$Load = \frac{load\_HB + load\_report + load\_SIP}{\#trans_{total}} \quad (4.1)$$

The SIP load is directly proportional to the total number of SIP transactions, therefore:

$$Load = \frac{load\_HB + load\_report}{\#trans_{total}} + \frac{\alpha \cdot \#trans_{total}}{\#trans_{total}} \quad (4.2)$$

Since heartbeat and report loads are fixed for a given set of fault tolerance settings,

$$Load = \frac{\beta}{\#trans_{total}} + \alpha \quad (4.3)$$

Equation (4.3) confirms that the load per transaction is inversely proportional to the total number of SIP transaction. Equations (4.4) and (4.5) show how the results from a single test can be used to calculate α and β and, then, extrapolate the load per transaction for any SIP traffic level—for a given set of other input metrics.

$$\alpha = \frac{\#trans_{success} \cdot \left(\frac{SAT}{RTT}\right) + \#trans_{failed} \cdot (\max\_FO + 1) \cdot (\max\_retrans + 1)}{\#trans_{total}} \cdot size\_SIP \quad (4.4)$$

$$\beta = \#HB\_msg \cdot size\_HB + \#report\_msg \cdot size\_report \quad (4.5)$$

Figure 4.3 depicts how the load per transaction varies with the number of SIP transactions for the example where α = 10 and β = 450,000.

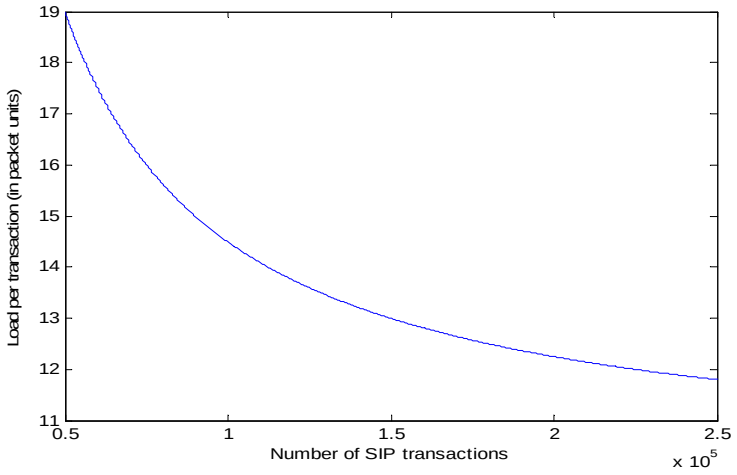


Fig. 4.3, Load per transaction vs. number of SIP transaction

Fault Model Parameters

The fault model greatly influences all results. The fault-related factors that can be tuned in the model are:

PER affects all communications but the consequences of packet losses change for each type of message lost:

- When heartbeat requests or responses are lost, the NS suspects PEs to be unavailable while they might not be—known as false alarm, or false positive—and in turn generates inaccurate PElists. Erroneous PElists are expected to lower the probability to selecting a currently available PE; hence, dependability decreases as well when PER raises. Note that the effects of erroneous PElists are significantly lower when the server pool deploys many PEs and especially if there are even more PEs than necessary to reach the maximum number of failovers.

- The impact of lost reports between the NS and PUs is similar to the impact of lost heartbeats. After some time, PElist become stale and do not reflect the current server pool status anymore—especially because in the model, the PUs take the PEs that failed to provide the IMS service out of the PElist. In the model, reports are sent on a best effort basis, i.e. there is no acknowledgement mechanism, so report retransmissions are not considered here. Another scheme could be envisaged where reports are retransmitted in case of reports are lost. There are two drawbacks to this solution. First, acknowledgment would add considerably more network load in the system. Second, retransmissions also increase the overall network load and delay the report update times.
- The loss of SIP messages triggers retransmissions. This has limited direct effects on dependability. PER is the probability that a message is lost, so the probability q that the SIP request or response is lost is:

$$q = 1 - (1 - PER)^2 \quad (4.6)$$

Typically, a SIP request is transmitted up to 7 times before a transaction is dropped, hence the probability Q that a transaction fails solely because of packet losses is:

$$Q = \left(1 - (1 - PER)^2 \right)^7 \quad (4.7)$$

PER is usually comprised between a few percents and 15-20% in wireless networks so the chances that PER alone causes failed transactions are low (e.g., for PER=20%, less than 0.1% of the transactions would fail because of packet losses).
Even though very few transactions are lost because of PER, it causes retransmissions that increase SAT. Retransmissions are triggered only when the timeout fires and then another round trip time needs to be added. This means that each retransmission delays the transaction completion time by the timeout duration, which increases with the number of retransmissions.
Higher PER also means higher load per transaction; each retransmission costs between 1 and 2 additional messages, depending on whether the PE has received the SIP request or not or whether the PE was up when the request reached it.

PE faults are expected to worsen all output metrics because PE unavailability (P_{OFF}) increases request failures that trigger retransmissions and lead to lower dependability, longer SAT and higher load per transaction.
Additionally, the mean time to repair (TTR) is probably at least as important as the P_{OFF} value. The reason is that unlike packet losses, PE faults are non-instantaneous events because of the node repair process (modeled by TTR), and the longer TTR, the more successive retransmissions. Thus, node crashes are more likely to increase the probability of transaction failures than random packet losses.
Because of the SAT definition it is difficult to predict how SAT varies with TTR. On one hand, the PE crashes cause retransmissions and, hence, longer SAT is expected when TTR raises. On the other hand, the longer TTR, the more likely retransmissions are

successive and dependability drops. This phenomenon pulls SAT values down—only successful transactions are used for SAT calculation.

RTT has effects correlated with those of TTR. It was explained in Section 4.3.3 that the transaction lifetime is proportional to RTT. If the transaction lifetime increases, the probability that an unavailable PE repeatedly contacted by a PU is repaired before the transaction is discarded increases as well. In conclusion, a bigger [RTT/TTR] ratio means higher dependability, but also longer SAT (cf. analysis in PE faults), less load (mainly because in the model the longer RTT, the less transactions).

Table 4.1, Summary of the parametric analysis for the system settings

	dependability	SAT	load	relevance
Load_SIP	0	0	–	0
PER_HB	–	+(?)	+/– (?)	+
PER_report	–	+	+/– (?)	+
PER_SIP	–	++	++	+++
P_{OFF}	–	+	+	++
TTR	– –	+/– (?)	+	++
RTT/TTR	+	+	–	+

Table 4.1 summarizes the preliminary analysis of the impact of the load and fault models, where the predicted variations of each output metric due to the increase of the corresponding input variable. For instance, when the number of packet losses affecting heartbeat (PER_HB) increase, it is expected that dependability moderately goes down, SAT will probably lengthen and it is difficult to project the effects on the overall load. Based on these observations, the relevance of each input variable is established, motivating that each set of fault tolerance settings should be tested against different PER, P_{OFF}, and TTR levels. Because of the large fault tolerance setting space, each input metric is set to only 2 levels.
Both PER and P_{OFF} vary between Low (1%) and High (10%). Each test scenario is systematically evaluated for the four input scenarios—so-called X_ticks—resulting from the combinations of PER and PE fault as shown in Table 4.2.

Table 4.2, Fault model settings used as input parameters for each test

	X_tick_1	X_tick_2	X_tick_3	X_tick_4
PER	L	L	H	H
OFF	L	H	L	H

The third and last system input variable expected to significantly influence the output results is TTR. Instead of using 3-D graphs, which would make them difficult to read and analyze, each test scenario is depicted with two independent graphs, one for each input RTT-related value.

The mean of the sum [TTF+TTR]—so-called cycle length (CL)—is the mean frequency of crash faults at a single PE. A given combination of CL and P_{OFF} permits to determine the mean duration of the failures resulting from the PE faults, which is shown in Table 4.3. In the model, CL is either fixed to 100 or 1000 seconds, so a fault approximately occurs every 1.5 and 16.5 minutes respectively. These values seem quite small because in real environments, faults are rarely so frequent; in some systems, certain failures even appear as rarely as once a month or every few months.
Longer CLs would model a wider range of real systems. Unfortunately, the simulated system operation time is fixed to 20hrs to keep the simulation time reasonable so, if CL values are increased, TTR and TTR are so large at the scale of the simulated time that there are not enough fault occurrences to properly get exponentially distributed ON and OFF periods with respective mean TTF and TTR. In order to 'compensate' the relatively short CL settings in the model, RTT is set to 100ms in all fault tolerance tests, which is also a small value for a real environment (unless all traffics are in a LAN).
Another option would be to scale down all the other time-related parameters – especially RTT, so that the maximum transaction lifetime is less likely to be close to or longer than TTR. Unfortunately, the problem is the same as with lengthening CL. Smaller RTT means more frequent state changes, i.e. longer simulation times for a given simulated operation time.

Table 4.3, Mean TTR for the different CL and P_{OFF} input values

	CL_1 (100s)	CL_2 (1000s)
P_{OFF} (1%)	1s	10s
P_{ON} (10%)	10s	100s

4.4.2. Reference Output Values – Standard IMS Scenario

In order to evaluate how well each fault tolerance setting performs, it is necessary to establish reference output levels from the non-replicated standard IMS scenario.
In this section, the standard IMS results are presented and analyzed in comparison with the foreseen effects of the environment input settings on the output metrics from the qualitative analysis made in the previous subsection.

Model Definition

In the standard IMS scenario, PUs request the service from a single S-CSCF. Since there is no replication implemented, the NS—as well as the heartbeat/report mechanisms—is no longer necessary and the network topology modeled becomes very simple, as illustrated in Figure 4.4.

The SIP traffic model, the S-CSCF and network fault models, and the SIP timeout and retransmission mechanisms are the same as in the RSerPool-based model described in Section 4.3.

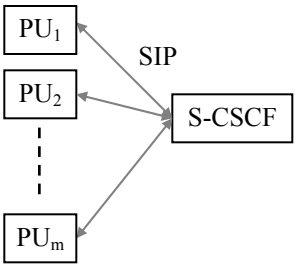


Fig. 4.4, Network topology of the standard IMS model

Results and Analysis

The standard IMS system is evaluated against the four X_tick input settings and the two CL settings, and three RTT levels. This way, observations can be made the influence of PER, the OFF probability, CL and RTT.
All the input test values are summarized in Table 4.4 and the corresponding results are depicted in Figures 4.5(a) and 4.5(b).

Table 4.4, Standard IMS test settings

	Test 1	**Test 2**
CL(s)	100	1000
RTT(ms)	100–200–500	100–200–500
InterSIP(s)	5	5
#PU	10	10

As planned in the qualitative analysis – and for all RTT settings – P_{OFF} has a significantly bigger impact on dependability than it has SAT and the load, while the opposite can be observed for PER:

- PE failures cause multiple successive retransmissions, and therefore worsen dependability. As proved by Equation (4.7), random packet losses cause isolated retransmissions that rarely lead to transaction failures on their own. Consequently, it is not surprising to observe such differences of dependability levels between $P_{OFF}(L)$ and $P_{OFF}(H)$, while the PER hardly makes any difference in terms of dependability.
- Successive retransmissions often lead to failed transactions so these retransmissions are not counted in the SAT calculation. Thus, SAT increases little with P_{OFF}. Isolated retransmissions hardly ever lead to transaction failures so they almost systematically lengthen the transaction completion time; the higher PER, the longer the average service access time of successful transactions.
- Since successive retransmissions are more and more spread out because of the exponential backoff retransmission mechanism, SIP request are (re-)sent less frequently during OFF periods than during ON periods. This means that for equal levels of P_{OFF} and PER, random packet losses affect a larger number of SIP requests than PE failures. This explains why the load increases much more with PER than with P_{OFF}.

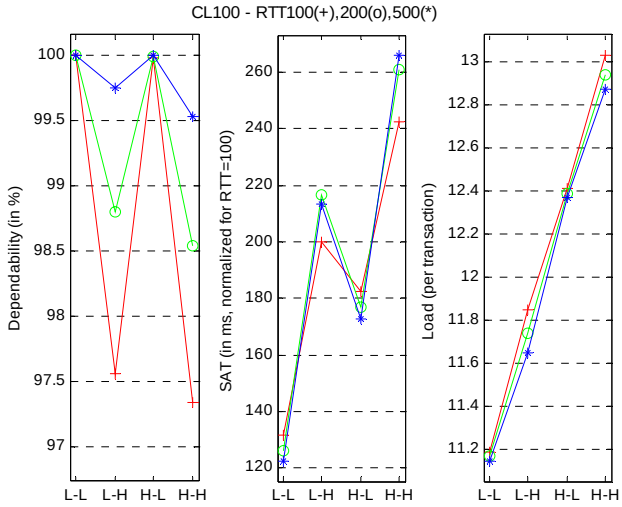


Fig.4.5(a), Standard IMS – *CL=100s, RTT=[100;200,500]ms*

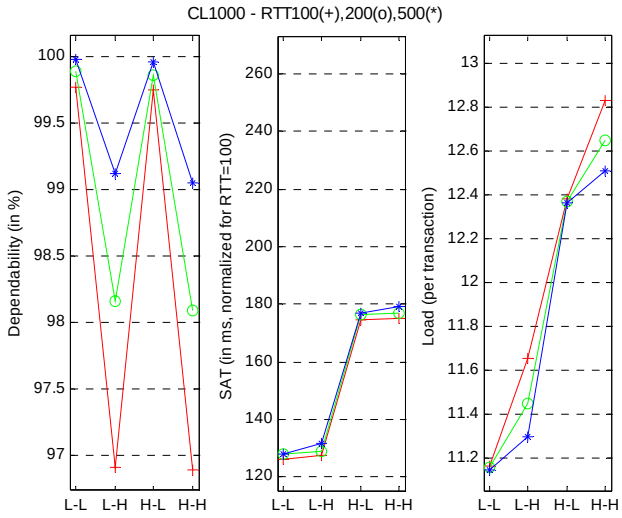


Fig.4.5(b), Standard IMS – *CL=1000s, RTT=[100;200,500]ms*

In both scenarios, the effects of longer RTT for a given CL are roughly the same.

- Increased RTT significantly improves dependability when the PE crash fault probability is high. When RTT increases, so does the maximum transaction lifetime and with it, the probability that the S-CSCF will recover before the transaction has failed. Hence, it can be consistently seen that for the exponential backoff retransmission mechanism and for a given CL, the longer the RTT, the higher the dependability. This confirms that the higher the [RTT/TTR] ratio, the more dependable the system becomes.
- SAT hardly changes with RTT for CL_{1000} and changes for CL_{100}, where it is about 10% lower for RTT_{100} as compared to the RTT_{200} and RTT_{500} scenarios. The fact that SAT slightly increases with RTT is explained by how SAT was defined: SAT

accounts for successful transactions only, so the higher the dependability, the more likely additional retransmissions have contributed to the increased dependability, hence, the longer the average transaction completion time.

- The load stays almost identical in the CL_{100} case. For CL_{1000}, the average load per transaction decreases for higher RTT values. The reason is that with longer maximum transaction lifetimes, fewer transactions fail. Therefore, more transactions trigger less than the maximum 6 retransmissions experienced by unsuccessful transactions; hence, less load per transaction.

Figures 4.6(a) and 4.6(b) illustrate the same results from a different angle; the goal is to highlight the effects of CL variations. For both RTT_{100} and RTT_{500}, similar conclusions can be drawn from the comparison of the CL_{100} and CL_{1000} curves:

- Comparing the respective dependability levels in the two figures further supports the observation that the higher the [RTT/TTR] ratio, the more dependable the system becomes. With CL_{1000}, dependability is always lower than in the CL_{100} case, i.e. dependability go down when TTR goes up—for a given RTT.
- The differences in terms of SAT for CL_{100} and CL_{1000} mainly concern the scenarios when P_{OFF} is high (i.e. 10%). This is because, in the CL_{100} scenario, TTR is shorter than the maximum lifetime—10 seconds average against [12.7; 63.5] seconds transaction lifetimes for RTT_{100} and RTT_{500}. In this scenario, retransmissions are quite likely to efficiently get around the PE failures. In the CL_{1000} case, when P_{OFF} is high (i.e. TTR=100s), retransmissions are not as likely to help keep the transaction 'alive' until the PE is repaired.
- The load is inversely proportional to CL. The reasoning is the same as for RTT variations, where 'RTT' should be replaced by 'TTR'.

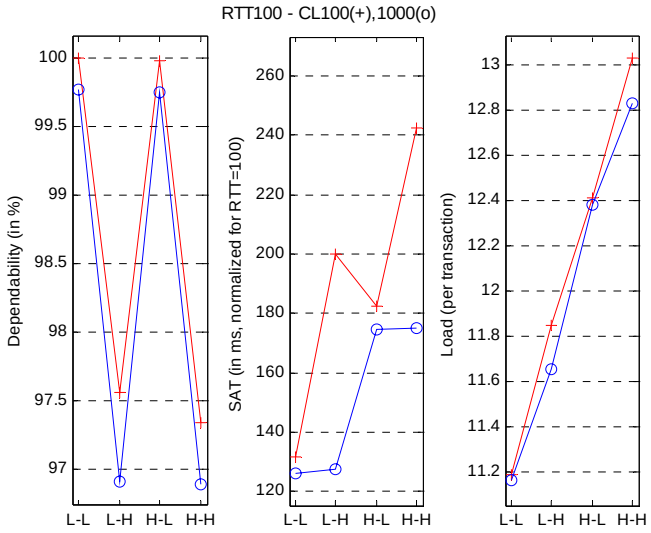


Fig.4.6(a), Standard IMS – *RTT=100ms, CL=[100;1000]s*

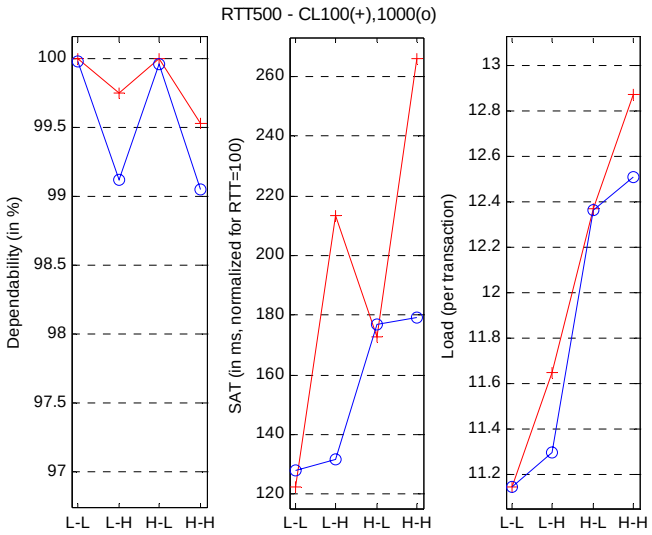


Fig.4.6(b), Standard IMS – *RTT=500ms, CL=[100;1000]s*

4.5. Fault Tolerance Configuration – Parametric Analysis

In this section, the main fault tolerance parameters of the RSerPool-based replicated IMS system are discussed and, for some of them, simulation results are added in order to confirm and determine the effects of these parameters more accurately.
Considering the large fault tolerance setting space that result from the parameter analysis, it is crucial to use a limited set of input system parameters so that the test space does not explode. By setting all the parameters discussed in Section 4.4 to 2 or 3 values each, the combination of all settings would add up to more than a million different tests. The following decisions were made in order to drive the design of experiment for the replicated IMS test scenarios:

- As we can see from Figures 4.5 and 4.6, both RTT and CL impact dependability and the load similarly and with relatively equivalent magnitude because of the RTT-TTR dependency. So both input variables are redundant in a way. Since CL variations change SAT much more than RTT variations do, it is preferable to compare the effects of the fault tolerance mechanisms on different SAT 'behaviors'. Hence, CL variations are maintained while RTT is fixed.
- Out of the three pre-selected values, the smallest RTT value (RTT_{100}) is chosen in order to have the shortest maximum transaction lifetime because of the short TTR settings that the model allow (see Section 4.4.1).
- For the same reason, the main focus is on CL_{1000}, but results for CL_{100} will be sometimes shown and analyzed when the scenario with frequent PE failures is relevant to specific fault tolerance parameter analysis.

4.5.1. Recovery Parameters

The parameters that directly affect the recovery process are:

- max_FO, the number of failovers;

- max_retrans, the number of retransmissions per PE before a failover;
- extra_PE, which is determined by the server pool size and max_FO. There must be a minimum [max_FO + 1] PEs in a server pool, so the server pool size in each test can be expressed as [max_FO + 1 + extra_PE];
- SSP, the server selection policy upon failover. This aspect has been extensively studied in [Bozinovski04b] so no further investigation is made in this work – the most dependable strategy, the so-called maximum availability SSP is assumed and implemented in the model.

Preliminary Analysis

In terms of dependability, failovers are expected to be effective against PE failures, while retransmissions to the same PE mainly help for temporarily long network delays and also for lossy communication links – even though failovers do this too. This means that, in general, dependability could be greatly improved:

- by increasing the number of failovers, and keeping the number of retransmissions low in comparison, when P_{OFF} raises;
- by increasing the number of retransmissions to the same PE, and inversely triggering fewer failovers, when communication delays and packet loss probability get bigger.

Communication delays and packet losses are modeled as random processes independent from the current system state. Therefore, RTT(msg_n) could easily be twice as short as RTT(msg_n–1) and RTT(msg_n+1) during the simulation, which is not always the case in a real system. Consequently, when RTT values temporarily increase, the exponential backoff retransmission mechanism probably has a smaller positive impact on dependability than it would have in correlated RTT scenarios.

The goal of deploying more PEs is to increase the overall availability of the pool. Therefore, larger pool sizes should positively impact dependability because PElists offer more PEs to pick from and therefore a higher probability to find available PEs to contact.

Since the timeout exponentially increases with the number of retransmissions, successive retransmissions become extremely costly in term of SAT. Also, every time a failover is operated, the retrans-counter is reset to ‘0’ and, so, the SIP timeout values stay low. Thus, it is expected that high max_FO and low max_retrans configurations provide the IMS service much faster than the opposite type of configurations simply because, for a given number of retransmissions, the sum of all timeouts is much lower in the ‘failover-aggressive’ approach than in the ‘retransmission-aggressive’ one.

Extra_PEs should not have any direct effect on SAT in the model. The SAT variations between the different extra_PE settings are more the consequence of different dependability outputs.

The overall load is made up of a fixed amount stemming from the fault tolerance mechanisms and a variable part due to the SIP traffic and that greatly depends on the number of retransmissions.

In the standard IMS scenario, it was found that the SIP load is lower for longer transaction lifetimes. Since failovers tend to reduce the maximum transaction lifetime, the overall load should also increase as compared to settings relying on many retransmissions.

Equation (4.3) showed how the fixed fault tolerance traffic contributes to the overall load for a given number of transactions per simulation run. The more PEs, the more heartbeats

exchanged with the NS—the amount of reports depends on the number of PUs—so the load is naturally expected to grow with extra_PE.

Tests Selection

Tests were run for many combinations of max_FO, max_retrans. For each combination, the total number of SIP request transmissions is given by:

$$\max\_requests = (\max\_FO + 1) \cdot (\max\_retrans + 1) \quad (4.8)$$

For each max_FO setting, the highest max_retrans is chosen so that max_requests is closest to seven, which is the standard SIP setting. In case two settings give two max_requests values equidistant from seven, the setting yielding the higher max_requests is picked. Table 4.5 shows the list of tests resulting from this choice, and the corresponding max_requests is indicated for each test.
The selected tests are all evaluated for three server pool sizes:

- 0 extra_PE, the pool deploys just enough PEs to allow all the failovers to be supported by different PEs;
- 2 extra_PEs, the pool has two PEs more than necessary for the failovers;
- 4 extra_PEs, the pool has four PEs more than needed.

Each test is referenced by a unique code that specifies the parameters values in this order: "max_FO/max_retrans/pool_size". For instance '3/0/5' refers to the test with 3 FOs maximum, no retransmission to the same PE before triggering a failover, and a server pool with 5 PEs (i.e. the 2 extra_PEs case).

Table 4.5, Summary of all the recovery settings tested and their corresponding max_requests values

		max_retrans						
		0	**1**	**2**	**3**	**4**	**5**	**6**
max_FO	**0**	*1*	*2*	*3*	*4*	*5*	*6*	*7*
	1	*2*	*4*	*6*	*8*			
	2	*3*	*6*					
	3	*4*	*8*					
	4	*5*						
	5	*6*						
	6	*7*						

Results and Analysis

In order to illustrate the tradeoff between number of retransmissions and number of failovers, Figures 4.7(a), 4.7(b) and 4.7(c) show three different configurations: the first configuration heavily relies on retransmissions, the second is a compromise between

retransmissions and failovers, and the third is all failovers. The input parameters and their settings are shown in Table 4.6.
For comparison, all test results are compared to the standard IMS setting, which is illustrated by the black line with diamond markers in all figures.
In the three tests depicted below, dependability, SAT and the load are always higher than the standard IMS levels, especially when P_{OFF} is high. E.g., dependability is consistently maintained above 99%. This is not the case for all combinations of fault tolerance settings that are not shown here though; the results for all fault tolerance combinations selected in Table 4.5 can be seen in Appendix B.4

Table 4.6, Replicated IMS test settings – recovery strategies

	Test 3	**Test 4**	**Test 5**
CL	1000	1000	1000
RTT	100	100	100
PU	10	10	10
InterHB	5	5	5
HB timeout	5	5	5
InterSIP	5	5	5
SIP timeout	exp.backoff T_0=100	exp.backoff T_0=100	exp.backoff T_0=100
max_FO	1	3	6
max_retrans	3	1	0
extra_PE	0–2–4	0–2–4	0–2–4

The results show that for the given network characteristics and PE fault model it is preferable to favor failovers over retransmissions in order to get the highest dependability levels in all input fault scenarios. The advantage of using many failovers instead of retransmissions-per-PE is that the timeout value always stays low despite retransmissions, so SAT becomes much shorter as well.
Modeling correlated network delays might reduce the differences between the configurations in terms of both the output dependability and SAT by permitting to fully make use of the exponential backoff mechanism when more retransmissions are used. At the same time, modeling correlated PER would probably amplify the differences between recovery configurations; unless most packet losses occur in PU's network, in which case it is likely that the request never reaches the PE, no matter which PE is contacted.
We can see from Figures 4.7 that the load slowly increases with the number of failovers. Looking in detail into the results revealed that the SIP contribution is roughly the same in the three cases and is similar to that of the standard IMS scenario. What causes such difference is the total number of transactions generated during each run, which was proven to impact the overall load inversely proportionally (cf. Equation 4.3).

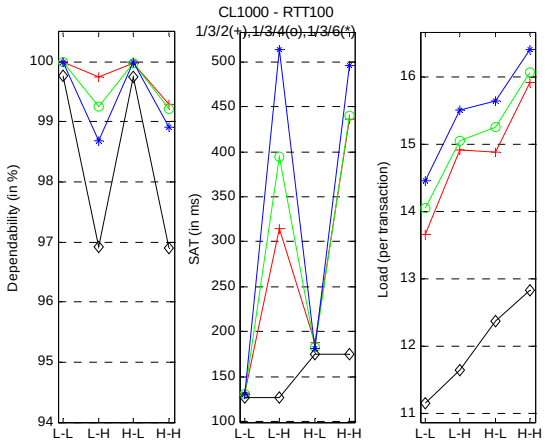


Fig.4.7(a), Replicated IMS – *max_FO=1, max_retrans=3, extra_PE=[0,2,4]*

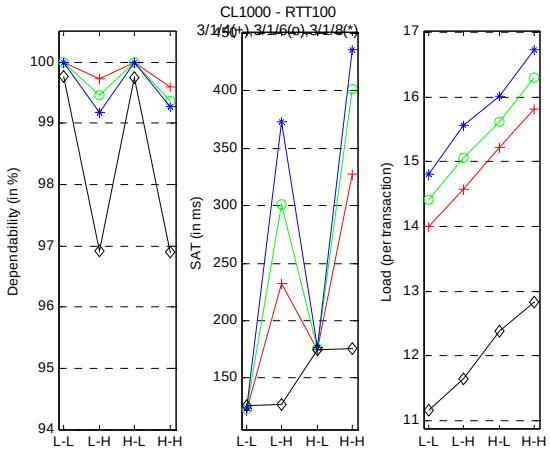


Fig.4.7(b), Replicated IMS – *max_FO=3, max_retrans=1, extra_PE=[0,2,4]*

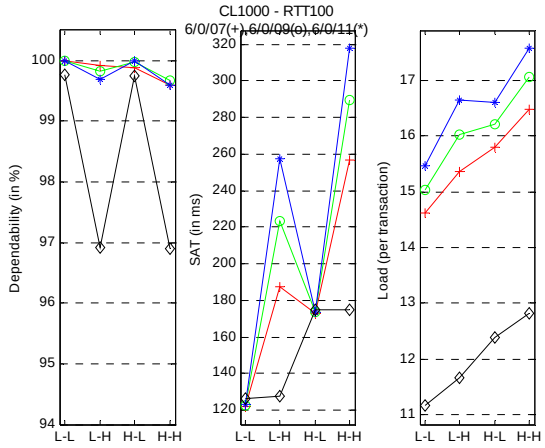


Fig.4.7(c), Replicated IMS – *max_FO=6, max_retrans=0, extra_PE=[0,2,4]*

It is surprising to notice for the fault tolerance configurations depicted, and most of the configurations tested in Appendix B.4, that the more extra_PEs, the lower dependability, especially when P_{OFF} is high. The reason is probably linked with a certain aspect of the SSP design, even though this cannot be verified from simulation outputs. Each PU deletes the identity of a PE after max_retrans unsuccessful attempts with this PE. When a PU has been through all the identities that are suspected by the NS to be available, it starts contacting apparently unavailable PEs on the list until a new PElist is received from the NS. Therefore, in scenarios with larger pool sizes, PUs are more likely to still have PE entries in their current PElist when receiving a new PElist, even when the previous PElist was never received or received very late because of longer communication delays. This means that in larger pool size scenarios, it is plausible that PUs send a request to 'low-ranked' PEs left in the lists soon—or just—before they receive a new PElist. These PEs have less chances to be available than the first rank in the new PElist received.

This characteristic should degrade the dependability of configurations with fewer failovers more because they offer fewer opportunities to recover from contacting the unavailable PE before the PElist is refreshed. Comparing Figures 4.7(a), 4.7(b), and 4.7(c) corroborates this assumption.

Also, when PER is higher, there are more instances when larger PElists at PUs become 'empty' as well and the behavior is closer in all pool size configurations, hence the dependability values are less sensitive to the pool size parameter for X_tick_4.

Results from the CL_{100} scenario permit to confirm the aforementioned reasoning. Figure 4.8 shows the output for a maximum of one failover and three retransmissions per PE. In the CL_{100} case, PE failures are shorter and, therefore, PEs recover more often as PUs send the last request before a PElist update. This explains why dependability levels—and thus SAT levels—are less sensitive to the pool size setting in the CL_{100} case than in the CL_{1000} one for all X_ticks.

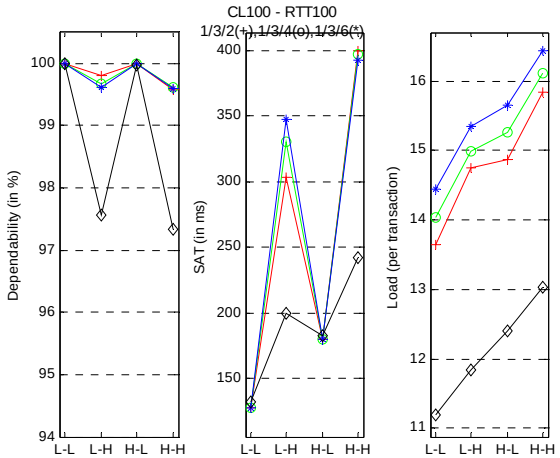


Fig.4.8, Replicated IMS – ***CL_{100}, max_FO=3, max_retrans=1, extra_PE=[0,2,4]***

4.5.2. Failure Detection Parameters

The accuracy of the feedback from the heartbeat mechanism can really help the PUs make the right choice when it comes to pick a PE for the first request of a transaction or for a failover. The heartbeat parameters are the inter-heartbeat (InterHB) time and the heartbeat timeout.

Heartbeat Frequency

Reducing interHB offers the advantage of refreshing the pool status image at the NS more often. Consequently, the fraction of time during which PElists are accurate is higher with frequent heartbeats, which should directly improve dependability. The drawback of higher heartbeat rates is the additional load that ensues.

Note that with the regular report technique implemented in the model, every PElist update at the NS should trigger the *immediate* distribution the corresponding report to all PUs. This report scheme has compounding effect on both dependability and load because:

- there is no delay between issuing the PElist and sending the corresponding report so the pool status information is made available to the PUs before it becomes stale (assuming short communication delays). Thus, the accuracy gains from more frequent PElist updates at the NS can fully benefit the PUs.
- when more heartbeats are generated, more reports follow, which further increases the load.

Three InterHB settings are tested. The five seconds InterHB time is compared to a less aggressive InterHB setting of ten seconds is used as well in order to evaluate if load-sensitive that cannot sustain frequent heartbeats. The three test settings are listed in Table 4.7.

Table 4.7, Replicated IMS test settings – failure detection strategies

	Test 6	**Test 7**	**Test 8**
CL	1000	1000	1000
RTT	100	100	100
PU	10	10	10
InterHB	5–10	2–5–10	2–5–10
HB timeout	5–10	2–5–10	2–5–10
InterSIP	5	5	5
SIP timeout	exp.backoff T_0=100	exp.backoff T_0=100	exp.backoff T_0=100
max_FO	1	3	6
max_retrans	3	1	0
extra_PE	0	0	0

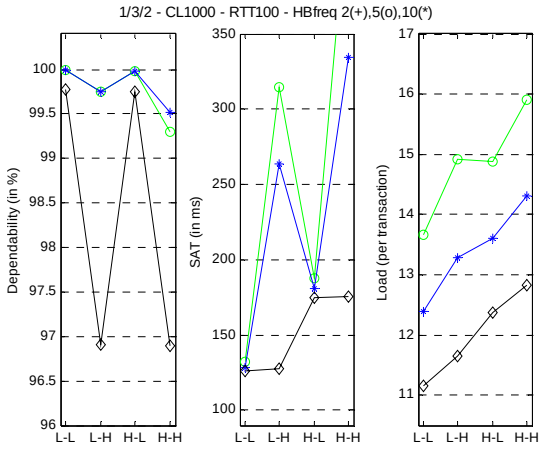


Fig.4.9(a), Replicated IMS – *1/3/2, interHB=[5,10]s*

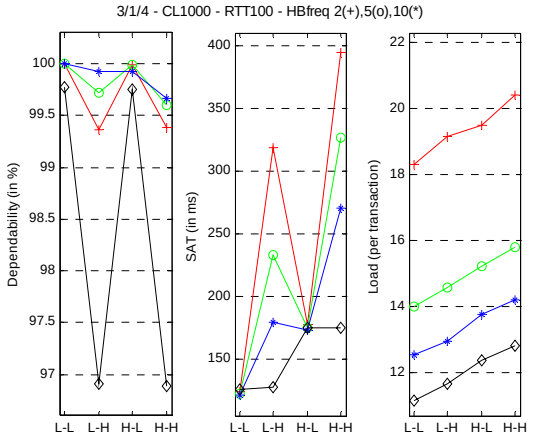


Fig.4.9(b), Replicated IMS – *3/1/4, interHB=[5,10]s*

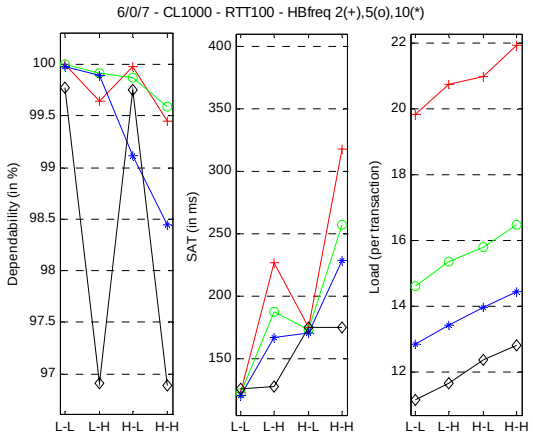


Fig.4.9(c), Replicated IMS – *6/0/7, interHB=[5,10]s*

Figures 4.9(a), 4.9(b) and 4.9(c) show that the effects of heartbeat frequency variations on dependability are similar to those of the pool size variations in Figures 4.7. Namely, when P_{OFF} is high, dependability sometimes becomes better when the heartbeat frequency slows down while the opposite is expected. Like for the pool size analysis, this influence is artificially caused by the conjunction of the SIP traffic model and SSP definitions when PElist. Varying heartbeat frequencies lead to the same discrepancy because, in the model, PElist updates and report broadcasts are synchronized so when PUs are sent reports less often, they are more susceptible to deal with empty PElists. Hence dependability is artificially raised, and SAT lowered, by slower heartbeat rates while the opposite influence is expected. Also, it can observed that the impact of this phenomenon diminish when increasing max_FO and/or PER, just as in the pool size analysis.
Nonetheless, the model permits to see consistently the positive effects of sending heartbeats less frequently on the overall load; e.g. by reducing the InterHB time by half, the gap between the standard IMS load and that of the replicated scenario is at least divided by two.

Heartbeat and SIP Request Timeouts

Playing with the heartbeat timeout is probably another way of gaining PElist accuracy. The heartbeat timeout determines how long before the pool status is updated at the NS a heartbeat should be sent. In the model tested, the heartbeat timeout is always equal to the InterHB value because the heartbeat sending and PElist update processes are synchronized. Instead, the heartbeat timeout could be shortened so that, when the PElist is updated, the heartbeat responses from the PEs are more recent; this provides another way to make the pool status information more accurate when PElists are populated by the NS. For instance, the heartbeat timeout could be set so that it is statistically longer than RTT is 90%. This aims at maintaining the heartbeat timeout as low possible while avoiding most false positive PE failure detections due to longer communication delays. This setting has not been tested.

Failure detection is also done at the SIP layer by implementing SIP request timeout. Results from tests 3, 4 and 5 showed that in the model the exponential backoff mechanism might not be so relevant. Therefore, a fixed SIP request timeout might lower the SAT levels without impacting dependability because, in the fixed timeout case, the second retransmission occurs twice as fast as with the exponential backoff, the third retransmission four times as fast, etc.
The same reasoning as for the heartbeat timeout holds when it comes to choosing the timeout duration so in the following tests, the SIP timeout value is so that it statistically encompasses at least 90% of the communication delays. Assuming exponentially distributed RTT, this probability is expressed as:

$$\Pr(request\_timeout \geq RTT) = \exp\left(-\frac{1}{RTT} \cdot request\_timeout\right) \quad (4.9)$$

Equation (4.9) gives $T_{90\%}$=230ms. The effects of using a different timeout strategy are tested as in Table 4.8.

Table 4.8, Replicated IMS test settings – SIP timeout strategies

	Test 6	Test 7	Test 8
CL	1000	1000	1000
RTT	100	100	100
PU	10	10	10
InterHB	5	5	5
HB timeout	5	5	5
InterSIP	5	5	5
SIP timeout	exp.(100)– fixed(230)	exp.(100)– fixed(230)	exp.(100)– fixed(230)
max_FO	1	3	6
max_retrans	3	1	0
extra_PE	0	0	0

The results in Figures 4.10(a), 4.10(b) and 4.10(c) consistently show that both SAT and the load are considerably reduced thanks to the fixed request timeout setting, while dependability worsens for fault tolerance configurations with few retransmissions and significantly improves for $P_{OFF}(H)$ when the number of failover raises. There are several reasons behind these observations:

- The fact that the fixed timeout value encompasses 90% of the communication delays greatly reduces the number of 'early' retransmissions that occur in the exponential backoff scenario where T_0 is set to the mean RTT, which explains why the load gets considerably closer to the standard IMS with the fixed value.
- The fixed $T_{90\%}$ setting does not help in scenarios favoring SIP retransmissions because the 'transaction lifetime per PE' becomes shorter, lowering the probability that a PE has recovered by the time the last retransmission to that PE is sent. For the test settings considered, i.e. T_0=100ms and $T_{90\%}$=230ms, the transaction lifetime per PE is almost the same for the second retransmission and becomes much shorter from the third retransmission in the fixed timeout case (920ms against 1500ms). When the number of retransmissions is below three, like in tests 7 and 8, the transaction lifetime per PE is actually larger with $T_{90\%}$ so dependability increases.
  Dependability improves in scenarios with fewer retransmissions also for the same the reason as for the load improvement.
- By avoiding some of the retransmissions caused by long communications delays, the SAT is reduced, especially for configurations with many retransmissions since each retransmission costs twice as much as the previous one in the exponential backoff scenario. With the 6/0/7 configuration, SAT is even about as good as the standard IMS case.

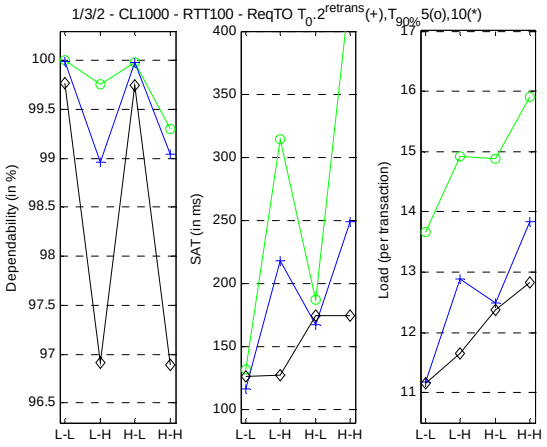


Fig.4.10(a), Replicated IMS – *1/3/2, ReqTO=[exp.T_0,$T_{90\%}$]*

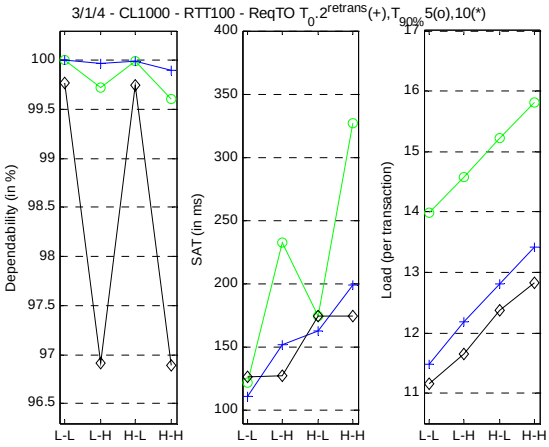


Fig.4.10(b), Replicated IMS – *3/1/4, ReqTO=[exp.T_0,$T_{90\%}$]*

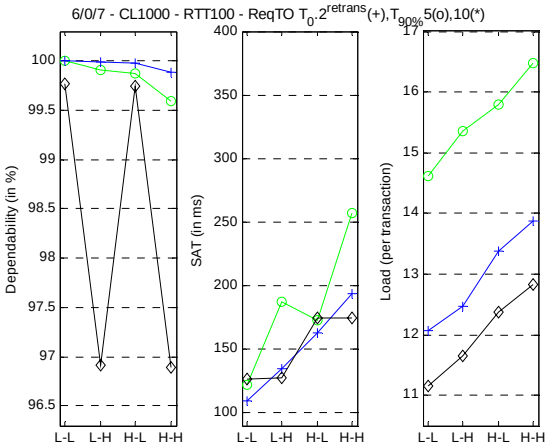


Fig.4.10(c), Replicated IMS – *6/0/7, ReqTO=[exp.T_0,$T_{90\%}$]*

4.5.3. Outlook on Report Schemes Analysis

Ideally, the accurate pool status should be available at each PU before it sends a new SIP request. The following schemes for pool status propagation from the NS to the PUs could be compared – but this is postponed to future work:

Proactive → the NS decides when to send reports to the PUs.

- Regular: the NS regularly broadcasts reports to all PUs, which has been thoroughly investigated in this work.
- Change-driven: the NS sends reports (or the relevant subset of the pool information only) when the pool status has changed during an update, i.e. when the status of at least one PE has changed since last heartbeat round. It is expected that for smaller pool sizes the pool status does not change often so it helps keep the report load low. The danger with this approach is that the one time each PElist might never be received by some PUs because of packet losses and this PElist is not sent again, only a different version will trigger the next report round – leaving some PUs with empty PElists longer than with the regular scheme.

Reactive → PUs request the latest pool status from the NS, and usually keep retransmitting the report request until they receive the report from the NS.

- Per-request: PUs request the latest pool status just before sending every SIP request. The advantage is to consistently provide the PUs with the latest pool status information generated by the NS. This solution presents two major drawbacks though. First, the report procedure delays the actual beginning of each transaction, impacting SAT that much. In scenarios with high PER, the report request would have to be retransmitted, which would increase SAT even more. Second, the report procedure is a two-way mechanism so load_report is proportionally larger. The overall load depends on the SIP traffic load.
- Cache: the PU caches the PElist for a given time period before it asks the NS for the latest version. This is almost equivalent to the regular scheme analyzed in the SAN model except that
  - The regular approach is on a best-effort basis, while the cache procedure is a two-way mechanism that makes it resilient to packet losses. This is not a problem though because the regular scheme could be augmented with acknowledgments from the PUs that would let the NS know which PUs might have not received a report. The NS could trigger the necessary subset of report retransmissions accordingly.
  - PUs request reports independently from the heartbeat process, so they are not sure they are getting recent pool status information, which seems less efficient in terms of failure detection

4.6. Model Application

The model helps understand and evaluate how server replication-based fault tolerance influences the IMS system both in terms of dependability and performance. For instance, the simulation environment could be used to validate analytical models that evaluate

dependability and/or performance metrics in similar network/communications/fault scenarios.
Also, the simulation environment can be used to determine which fault tolerance configuration would benefit a system most and at which cost. As it was shown, the setting space is very large so the general observations made from the simulation results about the influence of fault tolerance parameters can be reused to narrow down the range of tests to be conducted to find an optimal configuration.
In this section, several aspects of configuration selection are analyzed and an example based on the replicated IMS model is given to illustrate some of these aspects.

4.6.1. Configuration Selection Time

Design Time

Simulations are run for a given set of input parameter settings that can be drawn from system specifications or average values measured in the real system. Once the output metrics are evaluated for a subset of input variable settings and configuration settings, the selection criteria (cf. Section 4.5) should be tested in order to pick the single configuration that should be implemented the real system
This approach becomes limited for systems with dynamic fault models though. If the fault model varies with time, it is not ideal to pick the optimal configuration based on only one fault model (e.g. X_tick_n). One way to include this aspect in the offline process is to create aggregate input values. For instance, the overall PER value could be the sum of each PER level in a system times the individual fraction of time that the system experiences each PER level. The danger with this approach is that the output metrics have not been proven to vary linearly with the input variables so the configuration for the average PER, P_{OFF} and CL values might not be the one that gives optimal average dependability, SAT and load.

Run Time

In order to cope with dynamically changing fault models, the run time approach should be considered whenever possible. Run time fault-tolerance configuration tuning assumes:

- A database containing the results of the subset of configurations to be used for a given set of input variable values corresponding to the current system state.
- Real-time input metrics measurement techniques
- Protocol extensions in order to communicate the dynamic input values to the entities such as NS and PU that control the configuration parameters. E.g. the NS can adapt the reporting scheme (heartbeat timeout values, report frequency, report acknowledgements) according to the current PER and RTT levels for NS-PE communications; the PU can adapt the recovery strategy, e.g. by increasing the number of failovers when PE failures are long or adapt the SIP request timeout according to RTT (PU-PE communications).

4.6.2. Configuration Selection Criteria

Because of the evaluation approach, each configuration tested is evaluated with three output metrics. This makes it near impossible to determine which configuration is optimal by just looking at the graphs generated because it is very unlikely that a single

configuration can return optimal values for the three output metrics simultaneously. Therefore, some criteria are necessary to guide the selection process.

Output Metric Thresholds

The simplest way to rule fault tolerance configurations out during the selection process is to set a threshold value for each output metric. These thresholds should translate some of the system requirements. E.g. users might demand that dependability should be above 99%; the QoS requirements from an end-user application (e.g. online auctions) impose that SAT is below 300ms; the bandwidth of the system can only support a maximum load of 15 packet units per transaction.
Note that this technique might still leave several configuration candidates for the final choice. If one of the metrics has a higher priority than the other two, such as dependability would be in safe-critical systems, only two output metrics are bounded and the configuration that gives the best level for the third metric is picked (cf. example in Section 4.6.3).

Score Function

If no limit is imposed on less than two output metrics, a score function is needed in order to rank each fault tolerance configuration by returning a single score value for a given combination of output values. The score function is made up of contribution factors (CF)—usually one CF per output metric for which a threshold is not required. A contributing factor can simply be the ratio between the specific configuration output level and the standard output level. Note that for the dependability contributing factor definition it is more relevant to consider the ratio between undependability because:

- the difference between standard and replicated IMS dependability levels can be so minute that the magnitude of this ratio is much lower that of the SAT and load ratios and, thus, insignificant in the score function;
- this way, the variations of each contributing factor reflect on the system behavior similarly: $CF_{undep.}$, CF_{SAT}, CF_{load} increase/decrease when the system behavior worsens/improves respectively.

When one threshold is defined, the score function is calculated for the subset of configurations that respect the threshold requirement.

In Table 4.9, some score function examples are given. The first example does not favor any output metric for the selection process but the next two functions are shaped so that CF_{undep} variations have a bigger incidence on the final configuration rankings.

Table 4.9, Score functions examples

Score function	CF priority
$CF_{undep.} \cdot CF_{SAT} \cdot CF_{load}$	Fair
$\exp(CF_{undep.}) \cdot CF_{SAT} \cdot CF_{load}$	$CF_{undep.}$
$CF_{undep.} \cdot (CF_{SAT} + CF_{load})$	$CF_{undep.}$

4.6.3. Selection Examples

Let us use the simulation results discussed earlier to exemplify the selection process. The set of configurations considered for the following examples is restricted to the whole set of failovers and retransmissions per PE listed in Table 4.5. with the following settings:

- CL_{1000}, RTT_{100}
- Exponential backoff SIP retransmissions, T_0=100ms
- InterHB = 5s
- Extra_PEs = 0
- Fault model: X_tick_2 and X_tick_4 scenarios

The selection process is based on output metric thresholds and is repeated for two sets of requirements. Table 4.10 shows the selection criteria and the corresponding optimal configurations, which are found by comparing the graphs in Appendix B.4.

Table 4.10, Threshold requirements and optimal fault tolerance configurations

	Criteria 1	**Criteria 2**
Dependability	≥ 99%	≥ 99%
SAT	≤ 350ms	optimize
load	optimize	≤ 17 p.units
Optimal Config. (X_Tick_2)	2/1/3	4/0/5
Optimal Config. (X_Tick_4)	2/1/3	5/0/6

In the first example, dependability and SAT are bounded; therefore, the optimal configuration is the one with the lowest load among the configurations that meet the other two requirements. A compromise between failovers and retransmissions (2/1/3) to a same PE are best for both fault scenarios.
When a relatively large amount of load can be sustained by the system, good dependability can be achieved with shorter SAT than the 2/1/3 configuration when the PUs never send a SIP request twice to the same PE.

4.7. Conclusions

Summary

The standard IMS and RSerPool+IMS systems were modeled with SAN. These models were implemented in Möbius in order to evaluate the dependability/performance tradeoff and look for. Output metrics to look at fault tolerance with holistic approach in order to highlight the interdependence between dependability and performance.

The influence of input variables such as communication delays and parameters from the fault models was analyzed. Simulation results using the standard IMS model showed that:

- PER raises SAT and load, hardly impacts dependability (SIP messages);

- PE faults cause transaction failures, lower dependability, and SAT and load (the both because of the SAT definition);
- RTT/TTR ratio is very important because it determines how likely retransmissions can help against PE failures.

The analysis of individual fault tolerance mechanisms (recovery, failure detection) revealed that:

- The interdependency between dependability and load is the opposite of that with the standard IMS model; the more load, the better dependability. Here, fault tolerance mechanisms improve dependability at a fixed load cost, which depends on the failure detection and recovery settings. To significantly improve dependability, the additional fault tolerance load (introduced by heartbeats and reports) becomes much larger than the gains in terms of SIP load achieved thanks to the failure detection mechanism.
- Recovery and failure detection mechanisms rely on parameters that can be optimally tuned for a given fault model and set of dependability/performance requirement. For the replicated IMS—and for the specific fault and traffic models, and SSP tested—it was shown that
  - when the number of failovers increases (and the number of retransmissions per PE correspondingly decreases), both dependability and SAT improve but the load is greatly worsened;
  - the server pool should not have more PEs than necessary to execute the maximum number of failovers set for the current recovery configuration—when the server pool deploys extra PEs, all metrics get worse, especially for configurations with fewer failovers allowed;
  - the impact of the heartbeat frequency setting varies greatly for each recovery configuration so it is diffucult to draw general conclusions about its influence on the tradeoff. It is suggested that once the recovery configuration has been selected, a few heartbeat frequency settings are tested in order to optimize the output metrics for the given input fault model;
  - finally, having an accurate input model of the communication delay distribution permits to ideally set the SIP request timeout, which significantly improves the tradeoff, especially with recovery configurations favoring failovers over retransmissions per PE.

Directions were given to implement a selection solution at design time and at run time that indicates which fault tolerance settings optimize the dependability/performance tradeoff in a specific system for a given set of application requirements on the individual metric of the tradeoff. One example was used to illustrate how to manipulate simulation results to determine the optimal configuration.

Discussion about RTP

The high-level modeling approach used for the RSerPool+IMS system permits to quickly adapt the SAN model in order to get equivalent results and draw the respective conclusions for the RTP-like cluster solutions. The main difference is that in the RTP scenario the failure detection and recovery mechanisms are centralized directly within the

server set, which means that (1) there is no need for name resolution as in the RSerPool architecture (cf. Section 3.2.3 on Reports) and (2). This difference could be simply modeled by setting specifically low heartbeat and report communication delays and PER. This makes RSerPool look like it can be neither as dependable nor as fast as RTP. Nevertheless, it should be kept in mind that RTP is a very complex implementation because it relies on the additional cluster platform and shared database so much. This level of software complexity significantly affects RTP performance and dependability, which has been investigated in [Grønbæk07]. Therefore, it is not granted that RTP will outperform RSerPool in most environments.

5. State Replication and Consistency

When replicating stateful servers, the states held in these servers must be replicated as well. If the servers are deployed in distributed networks, state inconsistency may be introduced during the state replication process. Inconsistency can impact the system in many different ways, which are specific to the usage of the state information in relation to the service that uses this information: e.g. charging state inconsistency usually affects operators' revenues, while session state inconsistency might lead to incorrect transaction processing and, in turn, unavailability. Several consistency models have been designed that permit to address the inconsistency problem and are implemented via state dissemination, commitment and concurrency protocols. In addition to decreasing inconsistency, these solutions usually have a quite big impact on performance; hence newly introduced dynamic approaches attempt to optimize the tradeoff inconsistency-performance for some sets of inconsistency metrics, assuming that an accurate knowledge of the current inconsistency level in the system is known. In this chapter, a new inconsistency evaluation framework is presented that is made up of a few contributing factors that can be evaluated separately, either from real-time measurements or derived from the traffic model, and for a range of consistency models. A thorough discussion on IMS consistency shows the requirements on state replication in our system and the inconsistency evaluation framework is then validated for the IMS charging state.

5.1. Consistency Model in the IMS

Most of the time, inconsistency is referred to as state sequence disorder, also called event-ordering problem. With this approach, possible inconsistency definitions are such as the probability that a state update is ordered correctly, or the probability that all the state updates of a session are fully ordered, which is especially suited for dependent states, i.e. when state update n is a function of the state value resulting from state update n+1. Most common models corresponding to this notion are linearizability [Herlihy90] and sequential consistency [Lamport78].

These definitions and requirements originate from distributed computing, where all processes must have the same picture of the state (i.e. same data item) that they use as a unified input to their computational task. If the perception of the state differs in one process, then the logic is affected and it will not execute the correct operation.

In distributed communication networks, not all the replicated servers need to have the same state. For a given IMS session, only one copy of the IMS state is used for each transaction at the current master server; this means that at the moment when the master server processes the SIP request it does not really matter if the backup servers are consistent or not. The only requirement is that their state is consistent when it is needed, i.e. when there is failover to a backup server during the ongoing session and then, only

the selected backup server needs to hold a correct image of the state. Hence a state disorder can sometimes be transparent to the service/system: in our system, inconsistency can be observed—and, therefore, can have an impact—only when the state is read.
Not retrieving the correct state value leads to erroneous behavior of the system. Then, it is more relevant to focus on the correctness of the state when a read operation (RO) is done, which is called misread probability. This is defined in [Tanenbaum02] as strict consistency: *"Any read on a data item x returns a value corresponding to the result of the most recent write on x"*.
For instance, it is usual to read the state only once for billing purposes, at the end of the session, in order to deduct the amount of credits spent by the user from his account. In that case, a relevant metric is the distance between the state value read and the expected value, which measures the operator's losses. In the prepaid charging scenario, the operator wants to control the access to the network and potentially stop the session when all the user's credits are spent. Then, it is important for the operator to also have access to a consistent session state during the life-time of a session.

5.2. Inconsistency Evaluation Framework

5.2.1. Motivation

Dynamic, adaptive mechanisms have been proposed [Bozinovaki04a][Yu00] to restraint the inconsistency level under a certain threshold. For systems implementing such solutions, the need for accurate inconsistency evaluation is evident. However, inconsistency occurs in distributed systems, which makes it difficult to measure in reality (e.g. time stamps are difficult to use because of the clock synchronization problem). Here, we suggest an evaluation framework that uses the characteristics of a system in order to break the computation of the inconsistency level down to influencing factors that can be either measured or approximated from the traffic model and the description of the system. This evaluation approach offers the advantage of not requiring any specific, additional inconsistency evaluation functions to be implemented in the system.

5.2.2. New Evaluation Framework

One has to be careful when defining the correctness of the state. With SIP, the state at the server(s) where the read is being processed needs to be the same value as the one saved at the master server that committed the last state update. In other terms:

- The master server (local server) should replicate the state of interest to the backup servers (remote servers) before a read request is received by any of the remote servers.
- The state held at the local server, which is the most recent state information, should not be overwritten by older state information received from a server that was previously the master/local server.
- State update messages (SUM) are sometimes sent over unreliable links and therefore state ordering does not fully encompass all the causes for inconsistency to happen; the probability that a SUM never reaches a remote server(s) must be included in the evaluation.

Note that these requirements hold only if the state (or part of the state) of interest has been modified since the last read operation. This is because even if the last SUM generated is lost or disordered, the value returned by the seemingly stale state at the remote server is the same as the expected one and therefore the RO can be processed correctly.
The scenarios (shown in Figure 5.1) that lead to inconsistency in distributed communication networks are for the example of strict inconsistency, i.e. the probability to access a state that is not the last state committed in the system. The starting point corresponds to a read operation; the bold lines represent the paths to inconsistency instances, which occur when the last event of a path actually happens (shown in the grey text boxes). This evaluation approach can be generalized to any inconsistency definition that looks at the probability to read a correct state from the strict consistency point of view.

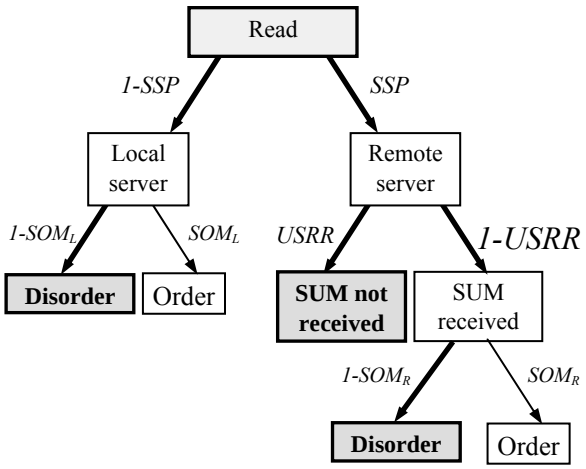


Fig. 5.1, Events sequentially leading to inconsistency, and their respective probabilities

Because of the server selection policy (or load balancing scheme) implemented in a system, a RO can occur at the server where the last state update was committed, the local server, or at another server of the system, a remote server. When a RO is done at a remote server, the state update message (SUM, see next section) carrying the last state update of the system must be received at this server; this message might never get to this server, especially for connectionless communications (e.g. over UDP), and the state would never be updated. Fulfilling only this requirement does not guarantee consistency: also, the latest SUM must be processed/committed at the remote server before the next RO arrives. The local server has already committed the last state update when the next RO arrives and, therefore, cannot be impacted by a SUM loss. However, if a SUM carrying the $n\text{-}1^{th}$ state update is received after commitment of local state update n, the correct value might be overwritten and corrupt the correctness of the state before the RO is processed. Usually a simple sequence number can prevent any local disorder to occur.
This discussion shows that the overall inconsistency depends on three factors:

- State Ordering Metric (SOM): the probability that the last state update in the system is committed at the server where the RO request is received, before this RO request is received. One should be careful when evaluating the SOM as it is expected to be different for the local server (SOM_L) and a remote server (SOM_R).
- Server Selection Policy (SSP): the probability of reading a state in a remote server. This probability depends on three system characteristics:
  - the server selection policy, which chooses the server where the next transaction will be processed;
  - the failure model;
  - the fail-over mechanism, which chooses the server where a retransmitted request should be sent to after a failure was detected (there are many existing policies that determine the destination server for request retransmissions).
- Unsuccessful State Replication Rate (USRR): the probability that the state replication is failed, i.e. that the SUM is not processed at the remote server, either because of packet loss or buffer overflow.

Those factors permit to devise a new measurement approach for the misread probability, directly derived from the probabilities illustrated in Figure 5.1:

$$Inconsistency = (1 - SSP) \cdot (1 - SOM_L) + SSP \cdot [USRR + (1 - USRR) \cdot (1 - SOM_R)] \quad (5.1)$$

5.3. Quantitative Inconsistency Evaluation

In this section, inconsistency in the IMS as defined in the previous sections is evaluated experimentally and compared to the new analytical approach in order to validate the evaluation framework. This work also gives the opportunity to foresee the expected inconsistency levels in the IMS.

5.3.1. Experimental System

We want to investigate inconsistency in the call control part of the IMS, whose experimental logical architecture is shown in Figure 5.2. We use the RSerPool example as state-sharing is not part of the standardization effort and therefore, this aspect needs to be investigated with special attention for this replicated architecture. More details about this experimental system are given in [Renier06].

In the RTP case, two options for state-sharing, or context management, are offered:

- the context management is distributed over the cluster and each server manages its own context manager, which in turn replicates the state to the other context managers on other nodes when an update occurs. This option is equivalent to the RSerPool case and similar conclusions will hold for both architectures
- the context management is centralized and the same context manager is accessed for all read operations, i.e. even from remote servers. This option ensures that there is no commitment or concurrency problem but it introduces a delay (to retrieve the state in the context manager) that impacts service performance more than the previous option. This will not be investigated specifically here.

The logical entities implement the IMS-like SIP call control servers (CSCFs) standardized by the 3GPP as defined in Section 2.3.1. The grey shaded entities represent the RSerPool components. The two redundant S-CSCFs form a server pool of Pool Elements and the P-CSCF was implemented as Pool User (see Section 3.2.4 for motivation). Every time a P-CSCF receives an INVITE request, it uses ASAP to request a name resolution from the name server in order to get the list of available servers to forward the requests to. The P-CSCF keeps this list in its cache for the whole duration of the session, a new session triggering a new name resolution request. The other communications during the session are made over UDP.

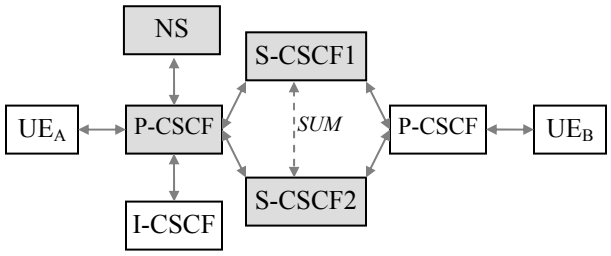


Fig. 5.2, Testbed logical topology for the IMS/RSerPool system. SUM is the message that contains the state information is sent from the local server to the remote server after every transaction completion

CSCF servers usually maintain a large number of session states simultaneously. In our example, the session state is only influenced by the transactions of its own session. Hence, parallel sessions are not required for the evaluation of inconsistency. UE_A follows a simple session/transaction generation pattern. Between the INVITE and BYE transactions, UE_A generates instant message transactions with the MESSAGE request. The inter-transaction time is the time between the moment when a transaction ends (completion or abortion) and the moment when the request for the next transaction is sent. In the testbed, the inter-transaction time is exponentially distributed, with mean value $1/\lambda$ set to one second. The server selection policy is round robin at the scale of the transaction, meaning that each request is sent alternatively to either S-CSCF, and the response(s) is sent back via the same S-SCSF (according to the SIP specifications, all messages in the same SIP transaction must be processed by one S-CSCF server only).
Because RSerPool does not specify any state-sharing functionality, we had to implement our own solution in the system. Our state-sharing mechanism is a best effort, message-based solution, over the connectionless transport protocol, UDP. When a transaction is completed at a server, the call state is updated in this server and the state update is replicated to its peer in a file sent with what we call a state update message (SUM), using the direct link between the two servers shown with the dashed line in Figure 5.2. The simplest models for state commitment and concurrency are used in our system. When the replicated servers receive the SUM, they extract the state information and immediately commit it, i.e. update the state. Also, when a read operation request is processed by a

server, the state is immediately accessed and no delay is introduced in order to ensure the correctness of the values read. The characteristics of the link (packet error rate, delay, etc.) between the replicated servers impact the time needed to propagate the SUM; their specific settings in the experimental system are given in Section 5.3.4.
There are no artificial server failures implemented in the testbed since the focus is not on the capability of the system to cope with failures of the SIP servers. The prototype SIP implementation used is not fully reliable though and failures can be observed at all SIP servers. Therefore, the fault-tolerant properties of RSerPool are used for fault recovery. When a failure/error is detected (timeout per SIP request set to 0.5 second), the P-CSCF retransmits the request to the back-up S-CSCF. The transaction is dropped if the timeout also expires when trying with the second server.

5.3.2. Measurement Approach

In this Section, we explain how we evaluated inconsistency (1) by implementing a solution in the experimental system, called experimental evaluation, and (2) by analyzing the influencing factors introduced previously, called factor evaluation. The goal is to compare the experimental and factor evaluations results in order to verify the validity of the proposed formula (c.f. Equation (5.1)).

We implemented an algorithm in the testbed to directly, and experimentally, measure the inconsistency level of the SIP fault-tolerant system. The (distributed) call state element in this example is a charging counter (*CSeq_server*) in the S-CSCF that keeps track of the cumulative number of successful MESSAGE transactions provided to the user. In this particular example, the approach to directly measure inconsistency is to also implement this counter in UE_A (*CSeq_msg*) and to communicate the UE_A's counter value to the S-CSCF in every SIP message.
When UE_A is aware that the last transaction is completed, i.e. it receives the corresponding final response, *CSeq_msg* is incremented before it is put in the next request UE_A sends. At the S-CSCF side, the local CSeq counter, *CSeq_server*, is saved with the call state. We consider that every request received by an S-CSCF initiates a Read Operation. Therefore, upon reception of a SIP request at an S-CSCF, we can be sure that the state is consistent at this server if the CSeq in the message is the next value in the incremental CSeq sequence, compared to the CSeq saved locally at the S-CSCF after the last state update (due to local update or SUM). In other words, there is inconsistency iff:

$$CSeq\_msg - CSeq\_server > 1 \quad (5.2)$$

When a SUM is disordered (meaning that the SUM arrives at the peer server after the request for the next transaction has been received) or when a SUM is missing, the value is not up-to-date and the RO increments the inconsistency counter (*InconCount*) at this server.
As described in the previous section, any server of the system can fail. When a transaction is unsuccessful because of a P- or I-CSCF failure, it might happen that the SIP messages are blocked on the SIP path before the state has been updated at an S-CSCF. Then, it would be unfair to the system to check inconsistency with a *CSeq* value that the

system potentially never saved at any of the S-CSCFs. Therefore, a RO can trigger an inconsistency check only after a successful transaction, i.e. only when we are sure that at least the local S-CSCF is aware of the last *CSeq* value generated by UE_A.
The inconsistency level is measured by dividing the inconsistency instance counter (*InconCount*) by the total number of ROs that requested the inconsistency check (*CheckCount*).
While rather simple to implement in this specific system, this experimental approach to evaluate inconsistency proves also to be quite limited. First, it requires dedicated code for inconsistency evaluation purposes, which may not be desirable e.g. for systems with low computational power. It also assumes that the state is monotonic and, therefore, relies on some sense of knowledge about the future state values, mandatory in order to assess the correctness of the state. Finally, UE_A is involved in the state evaluation, which seems against the operators' philosophy that suggests giving minimum control to the end users with respect to critical information, especially when dealing with charging-related information.

5.3.3. Factor Evaluation

In this section, we analyze how to evaluate independently each factor of the evaluation framework for the example of strict inconsistency requirement. Many inconsistency definitions/metrics can be broken down into influencing factors equivalent to the ones proposed here; then, those factors can be directly measured or analytically derived from the traffic model and system description.

State Ordering Metric (SOM)

In the system considered here, the state to be retrieved has monotonically increasing values (incremented by one for each new transaction); it should always be bigger than the previous one. Therefore, it does not make sense to update the state with a SUM when its *CSeq_msg* value is smaller than the current *CSeq_server* where it is received. That way, we prevent the propagation of disordered SUMs to several ROs. This would not be the case for non-monotonic state values (e.g. location information), which make it impossible to detect an out-of-date state value in the SUM and, thus, requires that the state is updated every time a SUM is received. Also, with this state update model, inconsistency cannot occur at a local server since an old SUM cannot overwrite the latest state at the local server, which implies that SOM_L is equal to one. Consequently, a disordered state update can only impact a remote server (SOM_R).
SOM_R is the probability that a RO request (i.e. any SIP request in our context) is processed by the remote server after the last SUM has been received and committed at this remote server. To evaluate this factor, we choose the moment when the state is updated at the local server as the reference starting point. Let (1) the Read Operation time (*RO_time*) be the time between the local state update and the next Read Operation, and (2) the State Update time (*SU_time*) be the time for the remote server to get and commit the SUM from the local server. Then, SOM_R can be expressed as the following probability:

$$SOM_R = \Pr\{RO\_time > SU\_time\} \quad (5.3)$$

Figure 5.3 shows all the delays that make up *RO_time* and *SU_time*, where: D_1 is the processing time between the final 200OK response leaves the local server and the corresponding SUM is sent to the remote server; *Delay* is the link delay between the two servers; D_2 is the processing time to commit the state update at the remote server after reception of the SUM. D_3 is the processing time at UE_A to complete the transaction after reception of the final 200OK response; D_4 is the inter-transaction time; D_5 is the processing time between the beginning of the transaction and the moment when UE_A actually sends the request; D_6 is the propagation time of the request from UE_A to the remote server. Note that the two other propagation times, namely between S_1-CSCF and UE_A, and between the two S-CSCFs, are negligible (the bandwidth is 1Mbps); only this propagation time was considered since non-negligible processing times at the P-/I-CSCF are to be considered in the calculation of SOM_R. D_7 is the processing time between reception of the read request and the actual checking of the state values.
Let us break those two times down into their respective delays:

$$\begin{cases} RO\_time = D_3 + D_4 + D_5 + D_6 + D_7 \\ SU\_time = D_1 + Delay + D_2 \end{cases} \quad (5.4)$$

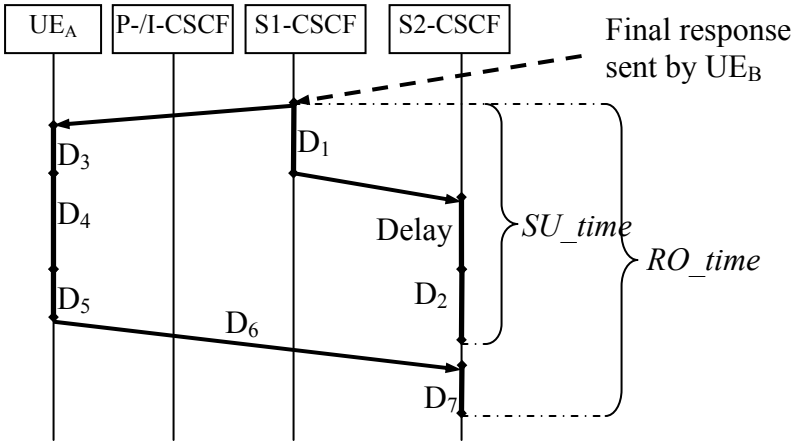


Fig. 5.3, *RO_time* and *SU_time* delays.

On one hand, according to the traffic model, D_4 is exponentially distributed, with mean value $1/\lambda$. On the other hand, we can assume that the other delays are deterministic because of the light load in the system. Therefore, in order to evaluate SOM_R, we can isolate D_4 and Equation (3) becomes:

$$SOM_R = \Pr\{D_4 > (D_1 + Delay + D_2) - (D_3 + D_5 + D_6 + D_7)\} \quad (5.5)$$

And in the case of exponentially distributed D_4, as defined previously:

$$SOM_R = \exp\left[-\lambda \cdot \left(\left(D_1 + Delay + D_2\right) - \left(D_3 + D_5 + D_6 + D_7\right)\right)\right] \quad (5.6)$$

Server Selection Policy (SSP)

Without failures and with the round robin server selection policy at the transaction scale, a new remote server is accessed for every successive transaction, i.e. *SSP* is 100%. Because of the inherent SIP failures, and because there are only replicated S-CSCF in the example scenario, the first S-CSCF contacted to process a transaction might be unavailable and the retransmitted request is finally processed by the local S-CSCF for the previous transaction, i.e. the RO is local; therefore, *SSP* is not equal to 1 anymore. We could have approximated *SSP* at 1, or derive it from the failure model. Instead, we measured it by comparing in the tcpdump files the server that processed transaction *n* successfully, noted *server(n)*, and the server that processed the read request for transaction *n+1*, noted *server(n+1)*. *SSP* equals the ratio between the number of cases when *server(n)* and *server(n+1)* are different and the total number of comparisons.
This illustrates that the proposed inconsistency evaluation framework can rely on different methods to determine its input values.

Unsuccessful State Replication Rate (USRR)

We assume that no state update is dropped due to buffer overflow since the traffic is very small compared to the memory and CPU capacities of the machines used in the system. Then, the probability that the SUM is not received is directly equal to the Packet Error Rate in the link between the two S-CSCFs.

Once the three influencing factors have been evaluated, they can be used in Equation (5.1) to directly derive the inconsistency level.

5.3.4. Results and Model Validation

Results

The previous section showed that, in our system, inconsistency is a function of the characteristics of the link used for the state replication, namely packet error rate and delay. These link characteristics were emulated at the S-CSCFs, in the function in charge of receiving the SUMs. To emulate delay in the link, the corresponding thread freezes for the desired time while the received SUMs are randomly dropped according to the chosen PER, before they are processed. The bandwidth was fixed to 1Mbps. We ran six tests with different values for those two link parameters, shown in Table 5.1. For each scenario, the duration of the test in terms of number of sessions established by UE_A is set to 1400 sessions. The number of messages per session is geometrically distributed with a mean value of 10; hence, the overall number of read requests to the session state (at which inconsistency is evaluated) is approximately 14000.
Although the system was lightly loaded, the measurements of the processing delays D_X (D_1, D_2, and D_3, D_5, D_6, D_7) showed that they were not deterministic and were distributed

with long tails. For example, D_7 was around 8ms for most of the RO requests but its values ranged from 5-6ms up to 800-900ms. Since the values in the tail of the delay distribution are not relevant for inconsistency, but they influence the mean strongly, we choose to use the most likely delay values of D_X (the modes of the empiric distributions) in the calculation of SOM_R.
Table 5.1 gives for each test scenario the final inconsistency results for the experimental evaluation (direct measurement of inconsistency) and the factor evaluation (via the right-hand side of Equation (5.1)), as well as the absolute difference between the two approaches.

Delay (ms)	PER (%)	**Experimental evaluation**	**Factor evaluation**	Absolute difference
20	2	**2.76%**	**3.80%**	+1.04%
20	2	**3.45%**	**3.92%**	+0.47%
0	15	**13.77%**	**13.38%**	-0.39%
300	0	**19.51%**	**23.38%**	+3.87%
300	15	**29.09%**	**33.77%**	+4.68%
300	15	**27.45%**	**33.62%**	+6.17%

Table. 5.1, Comparative results for the experimental and analytical evaluations of inconsistency

Analysis

The results show that the inconsistency level evaluated with the factor approach is slightly higher than the directly measured inconsistency in our experimental system in all scenarios except when there is no delay in the link used for propagating the state update (third row 3 in Table 5.1). When the delay is much larger, i.e. 300ms, the absolute gaps between experimental and factor results get larger as well: up to 6.17% in the worst case (delay=300ms, PER=15%).
The observation of bigger deviations in settings with higher Delay between the two servers can be explained via the empiric distribution of the delays D_X. As stated earlier, the delays that contribute to *RO_time* and *SU_time* are not deterministic and vary (sometimes even reaching large values), even though the load on the SIP/RSerPool system always stayed low. The variations in the D_X values appear to be a consequence of the SIP software implementation. Both *RO_time* and *SU_time* contain additive delay parts with such variation (the ones that are due to processing time); however, there are more of those in *RO_time*. As a consequence, the deterministic assumption used to compute SOM_R underestimates both *RO_time* and *SU_time*, but *RO_time* more strongly (contains four processing delays as opposed to only two in *SU_time*). Thus, the factor approach overestimates inconsistency due to the variation of the processing delays.
When Delay is null (i.e. resulting in smaller *SU_times*), the state is almost always updated at the remote server before the next RO is received, therefore *RO_time* is expected to be longer than *SU_time* and then, even longer *RO_times* do not hide potential inconsistency instances caused by disordering (only remaining cause of inconsistency is

packet loss). Hence, much higher accuracy is obtained with the factor evaluation in the scenarios with low or no Delay between the two servers.
Note that for the purpose of validation, we picked extreme settings for our tests that lead to very high inconsistency levels; it is rare to reach 15% PER, especially in the wired core network between two servers.

5.3.5. Framework Application Example

After discussing how to map the influencing factors to our IMS system, we concluded that SOM_L is equal to 1. Then, Equation (5.1) becomes:

$$Inconsistency = SSP \cdot [USRR + (1 - USRR) \cdot (1 - SOM_R)] \quad (5.7)$$

In Figure 5.4, we draw the inconsistency level as a function of the delay in the link between the replicated servers, for different given PER values; respectively 0%, 2%, 5%, 10%, and 15% from the upper curve to the lower curve. We assume SSP to be equal to 1 from the round robin setting, and that USRR is the PER. As stated earlier, when Delay is null, RO_time and SU_time are almost equal, and SOM_R is equal to 1. Note that in our system, for scenarios with no link delay between the replicated servers, inconsistency has the product of SSP and PER as lower-bound.

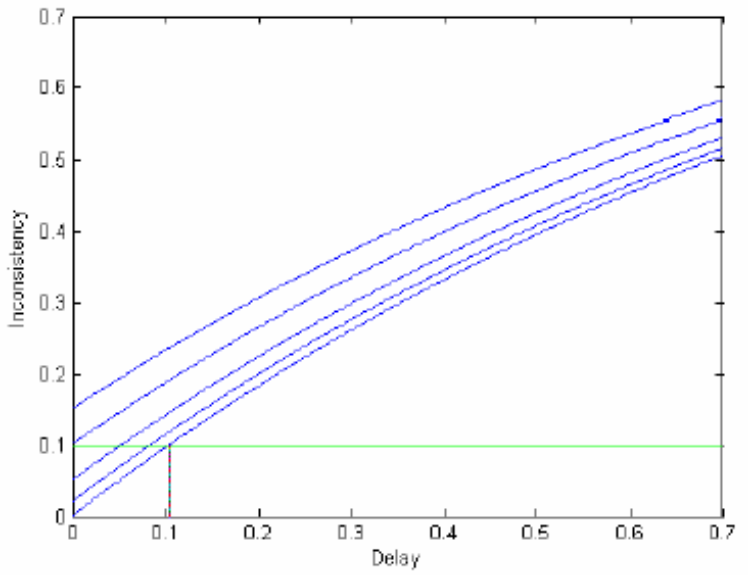


Fig. 5.4, Influence of Delay and PER on the inconsistency level. Each curve is the inconsistency level for a given PER (from bottom to top: 0%, 2%, 5%, 10%, and 15%)

The figure proves useful when analyzing the requirements on the system so that the inconsistency level stays below a given threshold. For example, it shows that no more than 10% PER (when no Delay) OR no more than 100ms Delay (when no PER) can be tolerated in our system to keep the inconsistency under 10%. Within those two bounding

ranges, all pair values Delay-PER allow inconsistency levels under 10%. This analysis highlights the need to consider the tradeoff between Delay and PER, which can be critical when designing a system. The type of protocol used to propagate the SUMs influences those two link characteristics and, consequently, influences also the inconsistency level. Reliable, TCP-like protocols ensure that fewer SUMs are lost when being propagated (lower PER) at the cost of longer delays while connectionless, UDP-like protocols offer lower delays but poor reliability to the SUMs. The impact of the protocol used can be analyzed analytically, based on its retransmission and congestion control models applied to the system of interest.

5.4. Conclusions

In this chapter, we introduced a framework for evaluation inconsistency in a replicated IMS. Direct measurements of inconsistency are deployed for model validation but in most cases they are not feasible in practical systems. The introduced new approach is based on an analysis of contributing factors, related to the Server Selection Policy, the packet drop rate between server replicates, and the disordering probability of state update messages. The first two factors are either known from system properties or can be estimated efficiently from the running system. The last factor is derived from parameters of the traffic model and different delay parameters in the system. The comparison of the evaluation approach via the contributing factors with a dedicated direct inconsistency measurement approach in an experimental prototype of IMS call control shows a close match despite simplifications on assumptions on processing delays. The relation between inconsistency and the contributing factors can also be used for network planning purposes.

PART II

—

Mid-Session Macro Handovers

6. Mid-Session Macro Mobility in the IMS

6.1. Introduction and Motivation

As discussed in the previous chapters, fault tolerance often relies on replication architectures that permit to shift the current load and communications of a failed entity to an available backup replica. So far, it has been assumed that the IMS functions could be replicated, and important aspects related to replication for the most critical function, namely the S-CSCF, were analyzed.

In some systems, replication is not a desirable option to cope with failures for several potential reasons:

- Replicating physical nodes and software can be expensive,
- Replication increases the overall load of a system, limiting its capacity,
- Replication can cause serious performance degradations.

Therefore, it is important to investigate scenarios where single points of failures are unavailable and block all traffic at an access network or IMS and discuss the potential options to mask these fault scenarios.

One trivial way to deal with timing failures perceived by the UE is the simple retransmission mechanism. Nevertheless, this assumes that the failed part of the system can be rebooted or replaced quickly enough to minimize the impact on the users' experience. When these two options do not work, the users lose/do not get access to the desired services either (1) because some access network failure prevents any communications with external network and isolate the users or (2) because the failure has happened at a non-redundant component, which makes the service unavailable. When this happens, connecting to another access network is the only active approach left not to lose the ongoing sessions. Also, in case of macro mobility, i.e. a handover that involves a change of access network, the QoS settings guaranteed in AN_{new} should be at least as good as those granted in AN_{old} before the failure so that the service keeps performing as expected.

Also, B3G systems will be characterized by a collection of radio and fixed networks providing access to IP-based services. Thus, providing mobility management across different access technologies will become a crucial task in B3G networks, not only for fault tolerance but also, e.g., for providing extended coverage or faster and/or cheaper connections.

6.2. Related Work and Problem Statement

While macro mobility is becoming an important feature of IP-based multimedia sessions as it is expected that mobile terminals will often switch between heterogeneous networks, operators want to control the access to their scarce radio resources by deploying access control and session control in their systems. The IMS can provide the desired control

levels but its current specifications do not allow for any change of the UE's IP address during an ongoing session; when a user moves to another access network (AN_{new}), all ongoing sessions have to be terminated and the corresponding session states are processed and then discarded. Additionally, the long SIP session setup procedures have to be performed once more at AN_{new}, which causes unacceptable disruptions for delay-sensitive, real-time services. Consequently, the standard IMS cannot provide seamless mid-session macro mobility.
Some enhancements have been recently proposed that reduce the disruption time during mid-session macro handovers in IMS environments [Larsen06a][Larsen06b][Castro06]. These SIP-based approaches (1) assume that the SIP session states can be kept (despite the session termination) and, then, use (2) context-transfer and (3) new signaling procedures in order to reduce the overall number of signaling messages that have to be exchanged for session establishment and resource allocation at AN_{new}. Even though these efforts achieve greatly improved disruption times, stateful applications running on top of SIP would still lose their session states and could not continue the service provisioning at AN_{new} where it got interrupted at AN_{old} without changing the implementation of both the SIP and application layers Lack of session continuity is not acceptable for stateful applications such as online gaming and auctions.
Ideally, application implementations should be kept as simple and standardized as possible, which helps the fast deployment of new services. So, another approach for macro handovers consists in using Mobile IP (MIP) instead. MIP is a network-layer protocol that provides session continuity to the upper layers by hiding UE's changes of IP address and therefore the session does not have to be terminated as the SIP layer is unaware of the change of IP address. Nevertheless, [Roos03][Faccin04][Chen07] have shown that several interoperability issues compromise the deployment of MIP in the IMS. These issues, plus some new issues raised in this thesis, are presented in detail in Section 6.5.1.

In this chapter, we first give some state of the art about macro mobility support in IP networks. Then, we highlight MIPv6 and SIP/IMS requirements, respectively, in terms of addressing scheme and their interaction with the application layer. Then, we analyze how these requirements are conflicting and lead to limitations when MIPv6 is used simultaneously with SIP in an IMS-controlled environment. Consequently, we introduce a MIPv6-based solution for supporting macro mobility in IMS that permits to respect the MIPv6 and IMS paradigms at the cost of a few additional non-standard operations and functionalities. We also discuss the impact of our solution on the standards, the implementation efforts that it implies, and the improvements that can be achieved.

6.3. Macro Mobility Protocols in IP Networks

6.3.1. Mobility Definitions

There is a set of mobility definitions that can be often found in the literature, e.g. [Schulzrinne00]. Here are the definitions used in this work.

Terminal Mobility

Terminal mobility allows a device to change its point of attachment to IP subnets, while continuing to be reachable for incoming requests and maintaining ongoing sessions across subnet changes.

- Pre-session mobility: the user device can be reached anytime at its current point of attachment,
- Mid-session mobility: the user can roam across networks, changing the point of attachment during an ongoing session.

Personal Mobility

Personal mobility allows a user to be contacted at multiple devices via the same logical address or to change the device used during ongoing stateless communications.

- Pre-session mobility: the user can be reached at several devices (but often times only one device is used once the user picks up),
- Mid-session mobility: the user can switch the ongoing session from one terminal to another.

Session Mobility

In addition to the two previous fundamental types of mobility, session mobility can provide mobility-related functionalities when UEs are on the move. This is not a 'core mobility feature' in the sense that it is not meant to make the UE reachable at the IP level, i.e. it does not deal with network layer connectivity. Nevertheless, session mobility is essential for stateful applications to handover as transparently as possible from the user's perspective.

Session mobility, also called service migration, is the ability to suspend an ongoing session and to resume it at another device with the same session state. This implies that the same settings should be allocated to the session after connecting to the new access point or changing the device. The user may want to move only a part of the session, e.g. in the case when a specialized device is more suited for handling one of the session media streams. This special feature is also referred to as component mobility or partial service migration.

Note that session mobility can be done only if session continuity allowed. Session continuity is ability of a system or application to maintain the session state despite mobility.

6.3.2. Macro Mobility Protocols Overview

Mobility solutions have been proposed that can be implemented at several layers of the protocol stack:

- Legacy networks (GSM, UMTS…) mobility is usually implemented at the link layer and is restricted to homogeneous networks, i.e. it allows the UE to change its point of attachment in the subnetwork. This means that the UE keeps its global IP address, which allows only for micro mobility (i.e. intra-domain handovers); E.g., in UMTS, micro mobility is within a given GGSN domain.
  Macro mobility has to be supported at higher layers.

- Network-layer mobility (Mobile IP, see Section 6.3.3) is provided for any kind of networks without regard to the link layer techniques deployed underneath and hides the mobility to the application by always showing a fixed global IP address.
- More recent research has brought mobility to the transport-layer, via mobility extensions of the newly introduced transport protocols, SCTP [Stewart 00][Riegel 06] and DCCP [Kohler 06a][Kohler 06b]. This solution moves the mobility support into the end nodes and keeps the network stateless.
- Mobility can also be implemented at the session/application layer with SIP (see Section 6.3.4). Moving the mobility management to the application layer means that mobility functions can be easily downloaded and installed on a device—the mobility software is needed at both ends though..
- Protocols originally designed for other purposes may support some aspects of mobility management. This is the case with RSerPool that completes mobile SCTP [Dreibholz03].

6.3.3. Mobile IP

MIP was first specified in [Perkins02] for IPv4 networks. MIP is intended to provide terminal mobility only.
A MIP handover consists of a movement detection and registration, called Binding Update (BU), with the Home Agent (HA). The UE, called Mobile Node (MN) in the MIP architecture, registers with a HA in its home domain. Each MN is given a permanent Home Address (HoA) in the home domain. When the MN visits a foreign domain, it gets a Care-of-Address (CoA) with DHCP or PPP. After it gets a CoA, the MN registers the new address with the HA. When a packet is received in the home domain, the HA forwards it to the MN through a tunnel using to IP-in-IP encapsulation, as illustrated in Figure 6.1. On the other hand, the MN communicates directly with the other endpoint, so-called Correspondent Node (CN). This leads to what is referred to as triangular routing.

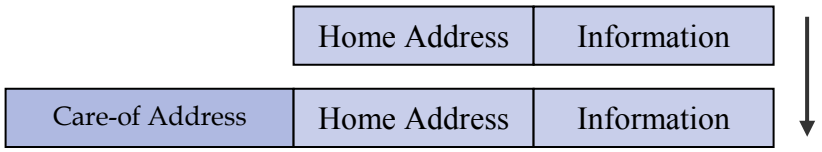


Fig. 6.1, Care-of Address encapsulation mechanism

The intrinsic problems of MIP are:
- Overhead due to IP-in-IP encapsulation, which impacts the overall application goodput. This is partly solved with header compression [Degermark99].
- Long process to detect movement (agent solicitation messages sent by MN after connecting to AN_{new}). Many novel solutions have been and still are proposed proactive mechanisms in order to reduce the handover latency (e.g. see [Feng04]).
- The MIP registration delays might be long if the MN and HA are far apart.
- Triangular routing introduces communication delays. To prevent this drawback, route optimization was designed to allow for direct communications from the CN to the

MN [Perkins01]. After receiving an a BU containing MN's CoA, the CN can start using the binding entry from its routing table to sends encapsulated packets directly to the MN.

With MIPv6 [Johnson04], the system benefits from the IPv6 neighbor discovery and stateless address auto-configuration for faster IP connectivity and movement detection. The route optimization support is implemented by MIPv6 by default but requires a new method called return routability procedure. It consists of two checks (i.e. four messages between the MN and the CN) to guarantee MN's identity, thus ensuring a secure BU at the CN.

6.3.4. SIP Mobility

SIP is capable of supporting all mobility types. Here we only detail the operations for terminal and session mobility, which are the required mechanisms for mid-session macro mobility in the IMS. To read about all the mobility types supported by SIP, refer to [Schulzrinne00].

Terminal Mobility

Terminal mobility impacts SIP at two stages: pre-session and mid-session.

- For pre-session mobility, the UE simply need to re-REGISTER in order to update its current location at the location manager (IETF) or HSS (3GPP).
- For the mid-session mobility, the UE sends a re-INVITE request to its correspondent. This request contains an updated session description with the new IP address, in the contact field of the SIP message. In addition, the UE has to register again so it can be reached for new incoming SIP messages.

Session Mobility

In the standard IETF architecture, the UE uses the REFER method to transfer the session to another device. This method is very similar to the re-INVITE and does not transfer session states to the new device.
Note that in the IMS, specifications do not allow session continuity at the SIP layer if the IP address of any participating endpoint changes, so the SIP session statea are lost and session mobility mechanisms can only achieve the equivalent of terminal mobility.

6.4. Scenario Description and Assumptions

Figure 6.2 shows the scenario considered in our work: a user equipment (UE_A) is attached to a UMTS access network and communicates with another user equipment (UE_B) attached to another IMS-based access network. We assume that a multimedia session, e.g. video conference, is ongoing between the two users. After some time, UE-A performs a handover from the old access network, AN_{old}, to the new access network, AN_{new}.
In case of stateful sessions, e.g. video gaming, session continuity is crucial since the users do not want to lose the history (i.e. the session state) of the ongoing session. Therefore, this work focuses on providing means for macro mobility support in IMS environments at the IP layer, which will hide the mobility to the higher layers and keep the session alive,

while respecting the standard resource allocation mechanisms specified at the session layer by the IMS.

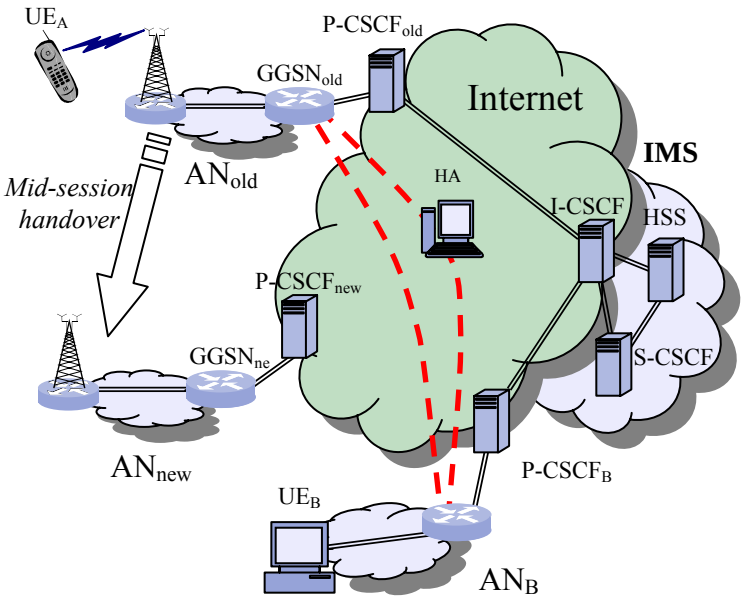


Fig. 6.2, Mobility scenario

6.5. MIP-IMS Interoperability Issues

In this section, we present the main problems related to MIPv6 integration in the IMS platform.

6.5.1. Delayed MIP Registration

Whenever a UE connects to a new access network (when switching the device on or after pre-/mid-session handover), it activates a primary context and gets its new IP address (CoA). The primary PDP context is designed to carry SIP messages only and, therefore, the access router will block other packet types such as MIPv6 registration packets; the MIP signaling is only possible after a secondary PDP context is created. This means that the UE cannot become reachable at the IP layer until a secondary PDP context is activated.

Before the secondary PDP context activation procedure can be completed, SIP messages have to be exchanged between the UE and entities in external networks for (1) SIP registration (S-CSCF) and (2) QoS negotiation (UE_B). Thus, SIP packets cannot be routed back to the UE and this issue leads to a deadlock situation where the UE can never complete either SIP or MIP signaling procedures.

6.5.2. Addressing Scheme Conflicts

MIPv6 hides UE's mobility to layers above IP and therefore requires that those layers use its HoA. On the other hand, IMS-based access control and QoS allocation negotiations are based on policies that are specific to the access network the UE is currently visiting, which is identified by UE's CoA. Those conflicting requirements impact two critical entities that participate to the access and service control, namely the P-CSCF and the GGSN.

P-CSCF

When the UE registers with the IMS, the P-CSCF checks for the address used in the SIP REGISTER request. As defined by the MIPv6 requirements, the UE should register its HoA at the SIP level. The P-CSCF will reject the SIP registration request because the HoA does not represent the current location of the MN, i.e. the HoA is not in the range of IP addresses matching the current access network. To avoid this problem, registering UE's CoA at the SIP layer is problematic because:

- this would require new interfaces between MIP and SIP at the UE so that the MIP communicates the UE's CoA to the SIP layer instead of the HoA;
- SIP would have to update the UE's location information after every handover –then it is similar to the SIP mobility case;
- changes of the IP addresses are not hidden to the SIP and application layers and, thus, session continuity cannot be ensured without additional implementation efforts.

GGSN

Once the UE is registered at the SIP level, it triggers a secondary PDP context activation for each data flow. In the process, the access router sets up the TFT and SBLP filtering functions, with the IP addresses of the data flow endpoints: UE_B's IP and UE_A's HoA. The addresses used to set the filters and the ones put in the IP header are not consistent in all scenarios of MIP communications between UE_A and UE_B, depending on whether the home agent is involved in the data path.

Based on these remarks, the following limitations can occur when MIP is used at the IP layer in an IMS environment:

- Without route optimization (i.e. reverse tunneling is implemented; the packets are systematically routed to UE's HA before being forwarding towards either endpoint):
  - Both GGSNs would block the packets originally sent by either endpoint because those packets are forwarded from the HA, which puts its IP address in the source IP address field of the outer IP header (TFT only lets through packets with either UE_B's IP or UE_A's IP),
  - Both GGSNs block the packets received from the endpoints: the SBLP does not allow packets with HA's IP in the destination field. The destination should be UE_B's IP or UE_A's IP, but reverse tunneling requires to send the packet to the HA first, i.e. HA's IP in the IP address header.
- With route optimization:
  - $GGSN_B$ blocks the packets from UE_A because the latter sends IP packets with its CoA in the source address field while the TFT at the GGSN is expecting the HoA,

- $GGSN_B$ blocks the packets from UE_B because the SBLP that has been set at the GGSN expects UE_A's HoA in the destination IP address field while UE_B uses the CoA.
- Incoming packets received from the UE-B are blocked at $GGSN_A$ by the SBLP because instead of UE_A's HoA in the destination IP address field, the CoA appears.

The address conflicts at the GGSN are also detailed in [Chen07] and summarized in Figure 6.3 for both the MIP bidirectional tunneling and route optimization modes.

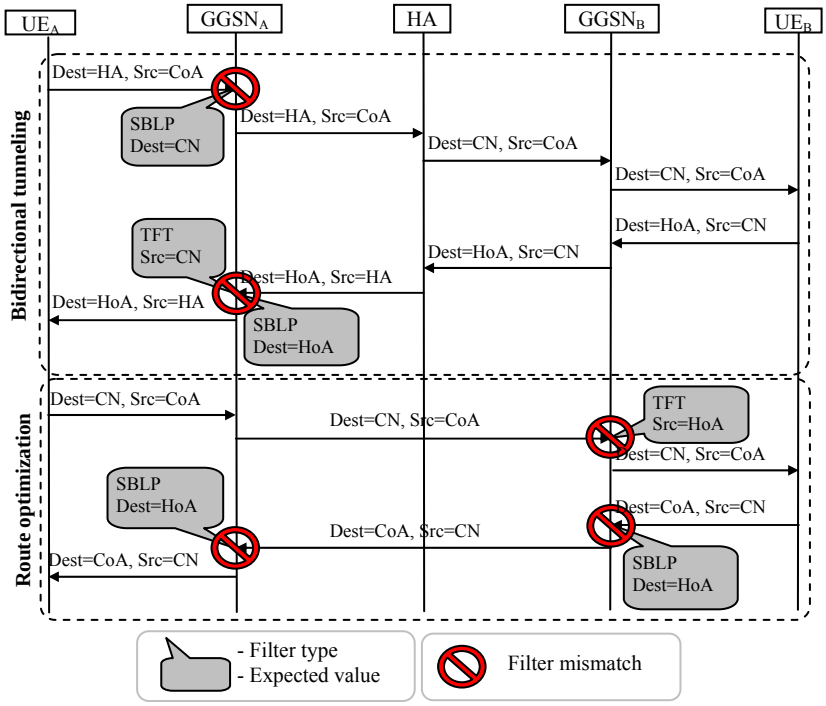


Fig. 6.3, Summary of address conflicts at the GGSN filtering functions

6.6. Solution for MIP-IMS Interoperability

We describe a solution that respects MIPv6 and IMS paradigms by adding functionalities to the access router and P-CSCF and exchanging a few additional MIPv6 messages.

6.6.1. Assumptions

The following assumptions on the system were made prior to designing the MIP-IMS interoperability solutions:

- The MIP HA is deployed in the Internet (it could be at the edge of the access network though) and, therefore, UE-A is in a foreign network in every access network from the IP point of view.

- Each access network is connected to a unique P-CSCF so macro mobility implies the change of P-CSCF as well.
- The IMS requires IPv6 for transport; so we assume that IPv6 is also deployed in the other domains considered in this work (Internet, UMTS core network, etc.).
- In terms of failure detection, UE_A is aware that the sessions are being dropped (forced by operator's policies) or that the communications are not reaching UE_B due to failures in its access network. If a communication path exists between the user equipment and the network monitoring entities, the latter can send a status report to the user equipment with the nature of the error/failure and a recommendation about the best recovery strategy; this is a network-initiated handover.
- Because of the scarce resources on the air interface, we assume that the resources in the Internet exceed those that are available in the access network. This means that we only need to focus on the resource allocation and control in the access networks. For simplicity, we also assume that the resources available to the UE at AN_{new} are at least as good as at AN_{old}. If that was not the case, the UEs would have to re-negotiate new QoS levels for the ongoing session and decide whether they want to continue the session with downgraded QoS.
- The two access networks that UE_A attaches to can belong to different operators. Roaming between different operators' domains presents inherent challenges [Roos03], especially related to security and QoS. We assume that the UE_A roams between access networks whose operators have signed roaming agreements, which establish clear rules about security (encryption, authentication methods and keys) as well as QoS compatibility, e.g. user profiles should be mapped to operators' specific Service Level Agreements (SLA).

6.6.2. Solution Overview

In order to allow the MIP registration procedure to take place as soon as UE_A has obtained a CoA, i.e. after a primary PDP context is activated between the UE and its current GGSN, the best solution is to allow MIP signaling through the primary PDP context. In general, it is reasonable to assume that the access router should treat MIP signaling as part of the signaling flow and not as a specific data stream. This is because (1) the high bandwidth available for the primary PDP context should benefit MIP mechanisms, (2) MIP signaling packets are not intended to any user application and should not be charged the same way as application data. So there is a rationale for including the MIP registration in the primary PDP context. For implementation purposes, one might argue that a primary PDP context can only carry SIP messages; in that case, a general PDP context should be activated instead since it is designed for mixed, signaling and data traffic. The same QoS allocation and charging policies can be applied to SIP and MIP signaling, while secondary PDP contexts are activated for the different data streams.
Both types of address conflicts introduced in the previous section can be solved with the same approach: by making the GGSN and the P-CSCF aware of both UE_A's addresses (HoA and CoA), both entities can look up in their binding table and prevent rejecting packets that contain a seemingly wrong IP address. This requires that both the GGSN and the P-CSCF are fully MIP-compliant –see next section for details.

6.6.3. Detailed Operations

This section presents the sequence of events that take place after a MN gets connectivity in a new access network. Those events are summarized in Figure 6.4. Note that this also holds when the user switches its mobile device on.

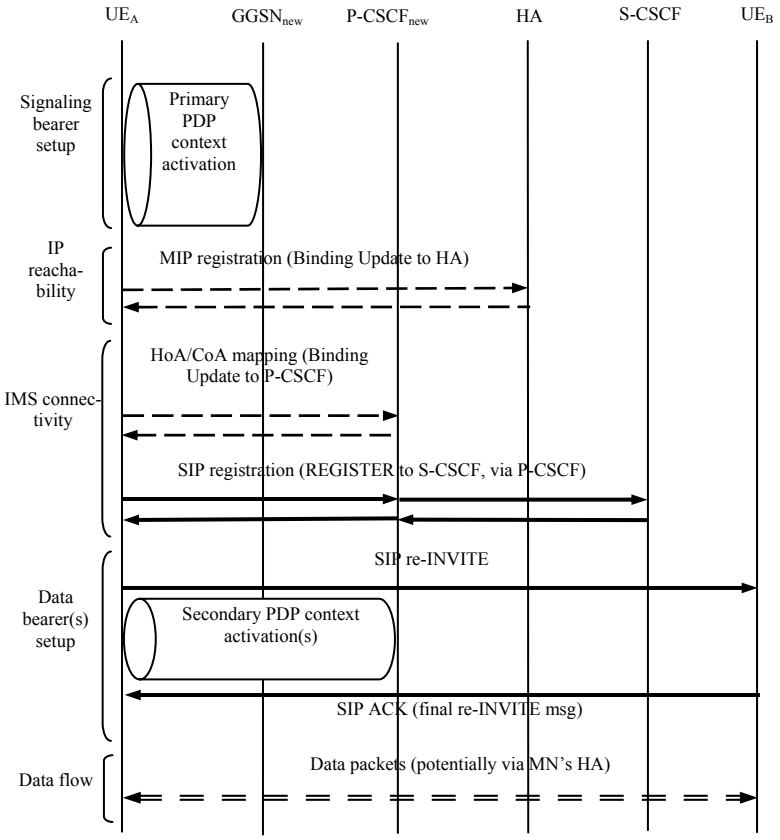


Fig. 6.4, Macro handover with MIP mobility support in IMS-based networks

After getting connectivity on the air interface in the new access network, UE_A activates a primary PDP context in order to get IP connectivity with external networks (IMS included). Assuming that MIP signaling can be sent over the primary PDP context—or a general PDP context—UE_A starts the MIP registration procedure immediately after primary PDP context activation in order to be reachable at the network layer by external nodes. When receiving the MIP registration request—namely, the binding update—from UE_A, the MIP-aware GGSN recognizes the message thanks to the Home Registration bit and the Mobility Header Type field in the Mobility Header. The primary PDP context now being improved with MIP signaling support, the access router does not discard the MIP request even though it is not a SIP message that is carried through the primary PDP context.

Then, UE_A needs to treat its current P-CSCF as a MIP correspondent node so that the latter knows about the HoA/CoA correspondence and does not reject the SIP registration request from UE_A. Concretely, UE_A sends a binding update to its MIP-compliant P-

CSCF, which updates its binding table. When the P-CSCF receives a SIP REGISTER with a forbidden address, it can look up the table for a mapping address that belongs to AN_{new}. Only then, can the P-CSCF process successfully the SIP REGISTER request with UE_A's HoA. Note that a new interface is needed so that SIP accesses the MIP binding information. After UE_A has registered with its S-CSCF, it activates a secondary PDP context (sending a SIP re-INVITE to UE_B) for each data stream that participated in the ongoing session(s) at AN_{old}.

In order to solve the address mismatch in the filtering functions at the GGSN, a similar address-mapping database to the one added to the P-CSCF needs to be implemented in the GGSN. For instance, the GGSN can extract UE_A's addresses from the MIP registration request sent to the HA before forwarding it; subsequently, the GGSN commits the address mapping when it receives the binding acknowledgment back from the HA. Another approach is to make the P-CSCF communicate both UE_A's addresses during the secondary PDP context activation process so that the GGSN sets the filters with those two addresses. Also, in Section 6.5.2, we showed that in bidirectional tunneling mode (between the two UEs) the GGSN blocks the packets that UE_A sends/receives to/from its HA. Thus, we suggest that the GGSN decapsulates the data packets to/from the HA in order to check for the validity of the destination and/or the source address in the inner IP header.

6.6.4. Analysis

Handover Times

In most scenarios, MIPv6 is expected to achieve shorter handover times than SIP mobility [Kwon02]. Nevertheless, in IMS-based networks, providing mobility support is not sufficient to perform a complete handover: access and service control functions have to be set at AN_{new}, this is the session mobility aspect of the handover. In the 3GPP specifications, the SIP INVITE transaction simultaneously provides location update – for mobility support – and enables the negotiation and application of QoS policies at AN_{new}'s GGSN with the activation of secondary PDP contexts – for access and service control. This means that in our scenario MIP mechanisms will be responsible for the terminal mobility at the IP layer and the SIP INVITE for the session signaling. Therefore, MIPv6 integration in IMS-based networks does not permit to shorten the handover delays when following the standard secondary PDP context activation after moving to AN_{new}. In the best case, MIP and SIP signaling (Figure 6.4) can be processed in parallel. The MIP registration procedures being shorter than the SIP ones, the MIP mechanisms do not impact the handover times, which are consequently expected to be the same as in the pure SIP-based handover case. Here, the only impact on the handover time could be the result of the computational load that the MIP mechanisms add to the system. This may slightly affect the overall performance of the system; e.g. the GGSN and the P-CSCF could slow down because of the additional processing and treatment of the MIP signaling.

Signaling Overhead

The additional MIP operations used in our solution consist of binding update procedures between UE_A and its HA, and its P-CSCF. Return routability procedure and binding

registration are made up of six small signaling messages (cf. Table 5.2 in [Fathi06]). So in total, our solution requires twelve MIP signaling messages.
Note that if we want to avoid the impact of triangular routing on the data traffic, an additional six messages are necessary to complete the whole binding update with UE_B.

Implementation Efforts

The integration of MIP in IMS-based networks is not straightforward and our solution requires some implementation efforts:

- The primary PDP context should allow for MIPv6 signaling. In the previous subsection, we argued why this requirement would not impact the IMS specifications much.
- The GGSN and the P-CSCF should be MIPv6-compliant so they process binding update messages and map UE_A's HoA and CoA at the IP layer. Both entities will be deployed in IPv6 networks and a particularity of IPv6 nodes is that they implement MIPv6 functionalities by default in their kernel. So, again, not much implementation costs are introduced here.
- A specific software architecture should allow the functions in the GGSN and the P-CSCF to resolve address conflicts by accessing the binding information for UE_A's addresses. In particular, an interface is needed for the higher layers to call the CoA-HoA address mapping functions in MIP and for MIP to respond to them accordingly. Cross-layer mechanisms [Srivastava05] could easily be an answer to this requirement.

6.6.5. Conclusions

The concepts that make MIP-IMS interoperability possible mainly rely on the standard MIP binding update registration at the P-CSCF and the HA. This is done at the cost of reasonable implementation efforts, especially between the MIP and SIP layers in the GGSN and P-CSCF. The UE, which implements MIPv6 by default as an IPv6 node, and the application do not need to be modified at all.
As compared to the original SIP procedures, our solution supports transparent session mobility. If the same feature had to be provided in the pure SIP scenario, interfaces between the SIP and application layers would have to be introduced so that SIP does not trigger the session termination and the application does not lose the ongoing session state as the UE's IP address changes. In this scenario, applications would have to be modified, which plays against one of the motivations for the IMS: easy and standard application development and deployment.

7. Enhanced MIP-based Mid-Session Macro Mobility

The analysis in the previous chapter showed that MIP does not permit to reduce the macro handover delays as compared to the original SIP in IMS environments because of the session mobility requirements. IMS macro handover optimizations found in the literature are SIP-based, which is not an optimal option for the reasons mentioned in Section 6.2. In this chapter, a novel mechanism for fast MIP-based mid-session macro mobility solution is proposed and the improvements that can be achieved, the implementation efforts that it implies, and its impact on the standards are discussed.

7.1. Solution Overview

During a mid-session handover, the most important aspect from the user's perspective is the disruption of the service, i.e. the discontinuity of the data flow at the application level. Therefore, emphasis should be put on creating a data path in AN_{new} as fast as possible in order to achieve a low latency handover. Ideally data could be carried in the first bearer created in the access network while controlling the resources allocated to this bearer. General PDP contexts meet this requirement since their particularity is to carry signaling packets and data packets, and to be granted resources by the access network.
In our scenario, resources have already been authorized and allocated in both UE_A's and UE_B's access networks before the handover. Thus, two arguments particularly motivated the design of an alternative solution to the complex end-to-end multimedia session setup: (1) after the HO, resources do not need to be dedicated to the ongoing session in UE_B's access network again, and (2) UE_A has stored media-authorization Our solution defines a new PDP context type, namely Mid-Session Handover PDP context (MSH PDP context). This new PDP context is similar to a general PDP but is activated with specific procedures detailed in the next section. The MSH PDP context activation process requires:

- A new interface between PCF/P-$CSCF_{old}$ and PCF/P-$CSCF_{new}$ so that PCF/P-$CSCF_{new}$ can get the session-related information that it uses to allocate resources for MSH PDP context and to charge the user appropriately;
- Additional information: the media authorization token for the ongoing session and P-$CSCF_{old}$'s IP address;

Our solution permits the data bearer to be set at AN_{new} without using SIP mechanisms (REGISTER and INVITE) and, therefore, to complete the resource allocation procedure much faster. After the unique data bearer is activated, UE_A uses MIP mobility mechanisms to re-establish IP reachability with UE_B and to start exchanging data packets.

Once UE_A's and UE_B's application layers can communicate again, UE_A transparently returns to the standard IMS-based operations (i.e. primary and secondary PDP contexts).
In summary:

- MIP and SIP procedures are decoupled
- MIP provides terminal mobility
- Session mobility is supported by context transfer
- Media resources are allocated with new PDP activation methods

7.2. Detailed Solution Description

Figure 7.1 shows the full message flow of our solution and will be explained in the following subsections.
Note that, even though it is not a 3GPP requirement, we assume that P-CSCF and PCF are collocated and can commonly access the states of the sessions they control.

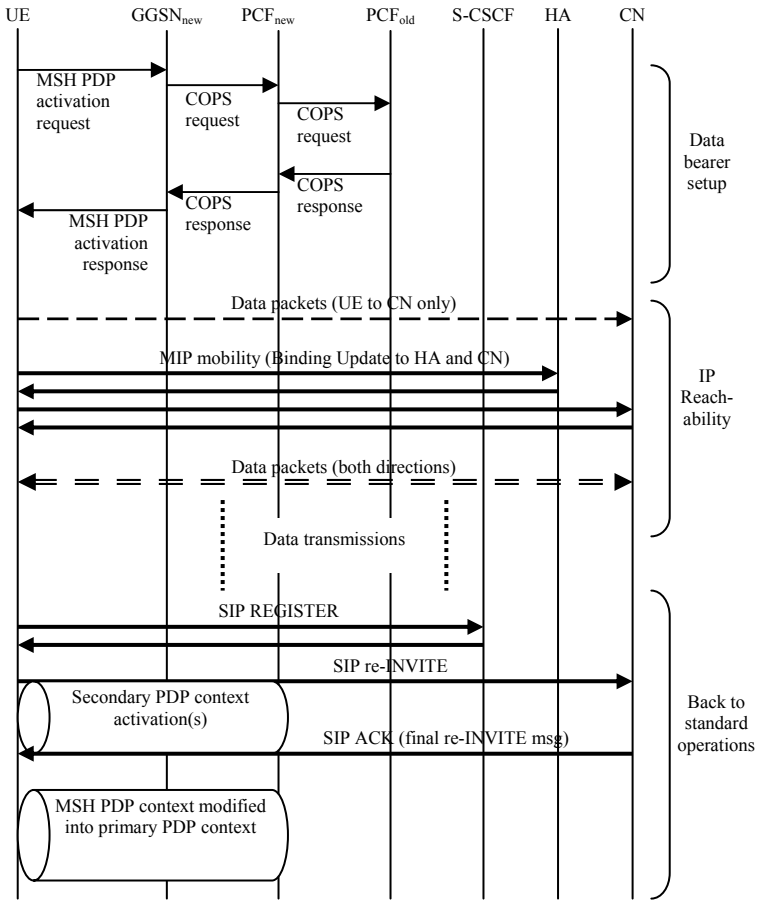


Fig. 7.1, Enhanced mid-session macro HO procedures

7.2.1. Data Bearer Setup

After UE_A looses connectivity at AN_{old}, it triggers the L2 Attach procedure at AN_{new} and authenticates with the network (UMTS AKA). When it detects that a session(s) was still ongoing before loosing connectivity at AN_{old}, the UE triggers the MSH PDP activation instead of the standard primary PDP context activation.
When setting the secondary PDP in AN_{old}, UE_A obtained a media authorization token, which specifies the PCF that generated this token. The token is added into the MSH PDP context activation request sent to $GGSN_{new}$. This request should be treated by the GGSN the same way as for a secondary PDP context so that access control for data packets can be set up as well. Therefore, $GGSN_{new}$ converts the PDP activation request into a COPS request that it sends to PCF_{new}. The latter behaves as a COPS proxy: it recognizes the specific MSH request and uses AuthToken to retrieve PCF_{old}'s location, where it forwards the request. In case AuthToken is insufficient to retrieve PCF_{old}'s location, e.g. when the two access networks are in separate administrative domains, then $P\text{-}CSCF_{old}$'s IP address is used to route the request to PCF_{old}. PCF_{old} recognizes AuthToken, accepts the request and looks into its database for the corresponding session information. It responds to PCF_{new} with a response that contains the user's profile, the SIP session state and the QoS levels authorized for this session in AN_{old}. When PCF_{new} receives the response from its peer, it (1) maps the old QoS levels into local levels – this depends on the roaming agreements between operators, (2) creates a temporary charging state and (3) interacts with $GGSN_{new}$ to finalize the activation of the MSH PDP context and open the access control filters.

7.2.2. IP Reachability

Once the MSH PDP context is activated, UE_A is able to communicate with external networks from AN_{new}. The priority at this point is to quickly inform UE_B about UE_A's new location so that the data packets can be routed properly from the CN to the UE. Note that the UE can start sending data packets after it receives the MSH PDP activation response because:

- UE_A already knows UE_B's IP address.
- UE_B will not reject those packets since MIP always shows UE_A's HoA at the application layer, which was knows by UE_B before the handover.

In our solution, the communications in AN_{new} are carried via a single PDP context, equivalent to a general PDP context. Signaling and data packets sent to UE_A are sent from different external nodes and make filtering policies difficult to implement for one PDP context. Therefore, we assume that the filtering functions that apply to the MSH PDP context are limited to checking for UE_A's addresses in the destination address of incoming packets (for the TFT filter to redirect packets into UE_A's MSH PDP context) and applying the QoS level agreed for the MSH PDP context.
UE_A should start the MIP procedures immediately after the MSH PDP is activated by sending a binding update to its HA and UE_B.

7.2.3. Back to Standard Operations

The goal of the MSH PDP context is to provide support in AN_{new} for IP multimedia communications as fast as possible but this PDP should not last until the end of the

session. This is because unless packet type differentiation can be implemented at the GGSN, the user would most likely be charged for signaling traffic at the same rate as data traffic when SIP packets are carried through the MSH PDP context. So whenever possible after the multimedia session is re-established, UE_A should trigger the necessary mechanisms to return to the standard operations defined by 3GPP, i.e. separate signaling and data traffic in primary and secondary PDP contexts. Nevertheless, UE_A should postpone those mechanisms when most of the MSH PDP resources are being temporarily used for the data stream(s), e.g., to download packets that had been buffered during the handover operations.

First, the UE sends a REGISTER to its S-CSCF via the MSH PDP context in order to:

- establish the SIP path between S-CSCF and $P\text{-}CSCF_{new}$;
- establish the security association at IMS level (IMS AKA);
- update HSS location database for future SIP requests addressed to the UE.

Because the mobility is handled at the IP layer, UE_B does not need to be updated about UE_A's new location information; on the other hand, the SIP re-INVITE is needed by the UE to obtain a new AuthToken in order to activate secondary PDP contexts at AN_{new}. After it forwards the final message of the re-INVITE (ACK message) to the UE, $P\text{-}CSCF_{new}$ logs the charging information related to the traffic in MSH so far and starts a new call state representative of the different charging rates of the data streams in their respective secondary PDP contexts. After receiving the final ACK message, the UE switches the data traffic to those PDP contexts and modifies the MSH PDP context to be dedicated to SIP signaling.

7.3. Analysis

7.3.1. QoS Resource Release at AN_{old}

The main goal of establishing PDP contexts is to allow operators to control their resources in the access network. When UE_A leaves AN_{old}, PCF_{old} and $P\text{-}CSCF_{old}$ need to keep the session information to be retrieved later on by PCF_{new}. Two scenarios are possible:

- $GGSN_{old}$ does not detect UE_A's movement and keeps the PDP contexts active while UE_A is moving to AN_{new}. After it responds to the COPS request received from PCF_{new}, PCF_{old} could release the resources in AN_{old} by sending a network-initiated PDP release to $GGSN_{old}$. Because it may happen that the COPS response from PCF_{old} to PCF_{new} is lost, $PCF/P\text{-}CSCF_{old}$ keeps AuthToken, the user's profile and the session state for a time $T_{retrans}$, set to 60 seconds, after responding to PCF_{new}. This way, PCF_{old} still has the information in case of COPS request retransmission(s).
- $GGSN_{old}$ detects that UE_A has detached and it releases its PDP contexts. Instead of deleting all session-related information when $GGSN_{old}$ requests it, $PCF/P\text{-}CSCF_{old}$ triggers T_{roam}, set to 120 seconds, and keeps the information for that duration. If PCF_{old} does not receive a COPS request from another PCF within T_{roam}, then it deletes the information. If it receives a COPS request, $T_{retrans}$ is triggered after sending the response.

7.3.2. Security Issues

IETF specifications require that the Media-Authorization header sent by the proxy (i.e. the PCF in our 3GPP UMTS environment) should be protected from eavesdropping and tampering [Hamer03]. In case AuthToken is intercepted anyway, it is recommended to set AuthToken timeout value to a few seconds to protect against replay attacks. In the mid-session handover case, we do not know how long the session will last before UE_A moves and, consequently, how long AuthToken lifetime needs to be. There are two solutions to this problem: (1) operators take the risk of replay attacks and allow for long AuthToken timeouts or (2) the PCF provides UE_A with new, additional specific token, that we call AuthTokenRoaming, which UE_A can use only when it activates an MSH PDP context in a new access network.

7.4. Quantitative Analysis

7.4.1. Assumptions and Methods

Here the handover times for both the original SIP and the enhanced MIP solutions are analytically compared. In both cases, the starting point is just after the UE has completed the L2 Attach procedures and is about to start the first bearer activation. Now we discuss for both mobility solutions when the handover would be considered completed:

- In the SIP case, an INVITE needs to be processed to allocate a data bearer for each media stream. So the handover finishes when the final response of the last INVITE transaction is received at UE_A. The SIP handover time is therefore proportional to the number of media streams to be created in AN_{new}.
- In the enhanced MIP case, all resources are available at AN_{new} as soon as the MSH PDP context is activated, which means that the MIP-based handover time is completely independent from the number of media streams that participate in the ongoing session moved to AN_{new}. The MSH PDP context activation is not enough from the UE point of view because it is important to consider E2E connectivity. Therefore, the MIP-based handover is complete when the BU with the CN/UE_B has been successfully completed.

The following analytical discussion abstracts the difference of packet size between SIP and MIP messages.

Looking at the message flows in Figures 2.6 and 7.1, we can list the different types of message exchanged and their number for the standard SIP and enhanced MIP mobility solutions, as seen in Tables 7.1 and 7.2.

Table 7.1, Message requirements for the standard SIP-based mobility

Operation	Messages	Number
primary PDP context activation	UE_A – $GGSN_{new}$	2
SIP REGISTER	UE_A – S-CSCF	4
secondary PDP context activations	$GGSN_{new}$ – PCF_{new}	2 (per stream)
resource allocations	UE_A – UE_B	11 (per stream)

Table 7.2, Message requirements for the enhanced MIP-based mobility

Operation	Mesages	Number
MSH PDP activation	UE_A – $GGSN_{new}$ $GGSN_{new}$ – PCF_{new} PCF_{new} – PCF_{old}	2 2 2
binding update (with HA)	UE_A – HA	2
binding update(with UE_B, (including the 4 route returnability messages)	UE_A – UE_B	6

Assumptions on communication delays in the access network and in the Internet have been discussed in [Kwon02] and [Fathi06]. The same assumptions are used in this work, and are shown in Table 7.3. GGSN-PCF delays are very short because the PCF is collocated with the P-CSCF, which, by default, has to be deployed in the same domain as the GGSN so few hops should separate them.

Table 7.3, Communication delays assumptions

Mesage	Delay
UE_A – $GGSN_{new}$ (when UE_A is in AN_{new})	20ms
$GGSN_{new}$ – PCF_{new}	5ms
PCF_{new} – PCF_{old}	100ms

In the discussion that follows, T[A–B] refers to the communication delay between any given node A and node B.
T[UE_A–S-CSCF], T[UE_A–UE_B] and T[UE_A–HA] are used as input variables.

7.4.2. Results and Analysis

Figure 7.2 represents the handover times for the SIP and MIP solutions together in each subplot. Each subplot is generated for a given T[UE_A–UE_B]—50,150,250ms—while T[UE_A–S-CSCF] is the input variable for the SIP curves and T[UE_A–HA] the input for the MIP curves. Both values are varied simultaneously and range from 50ms to 250ms.
For each T[UE_A–UE_B] scenario, the SIP handover times are calculated for an ongoing session with 1, 2 and 3 media streams respectively.
It makes sense to compare MIP ans SIP handover time for a fixed T[UE_A–UE_B] value because in a real setting, this delay would be the same whether the UE invokes the SIP or the MIP mobility solution.
These results show that MIP handover is always faster than the SIP handover when T[UE_A-S-CSCF] and T[UE_A–HA] are identical. It can be seen how much the number of streams in the session impacts the SIP handover time. The latter is also very sensitive to the delays between the UEs. This is because of the extensive number of E2E SIP

messages necessary to set the QoS parameters for each media stream in the session in both UEs' access networks.

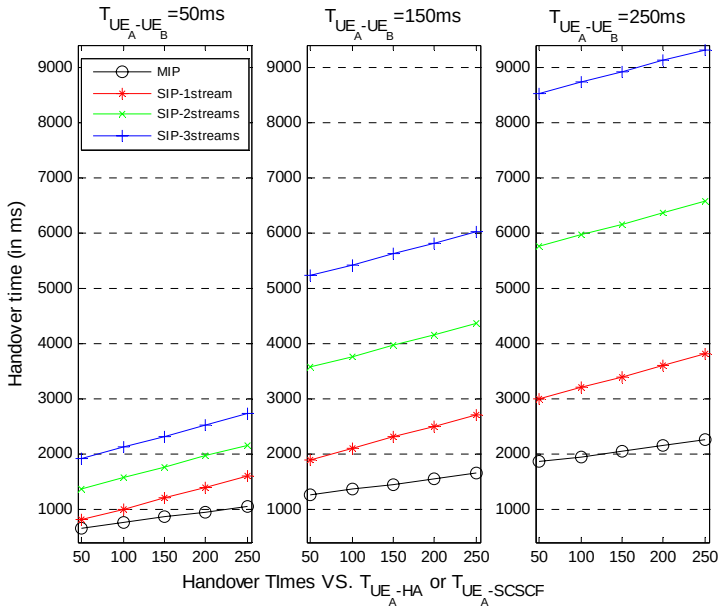


Fig. 7.2, SIP and enhanced MIP handover times

In a real environment, T[UE_A-S-CSCF] and T[UE_A–HA] would most likely be different. The graph clearly indicate that By comparing the handover time equations derived from Tables 7.1 and 7.2, it can be easily calculated under which conditions on T[UE_A–S-CSCF], T[UE_A–UE_B] and T[UE_A–HA] the MIP solution becomes slower than the SIP one, where n is the number of media streams in the ongoing session to be moved to AN_{new}:

$$T_{MIP} \geq T_{SIP} \quad (7.1)$$

$$2 \cdot T_{GGSN-PCF} + 2 \cdot T_{PCF-PCF} + 2 \cdot T_{UE-HA} + 6 \cdot T_{UE-UE} \geq 4 \cdot T_{UE-SCSCF} + 2 \cdot n \cdot T_{GGSN-PCF} + 11 \cdot n \cdot T_{UE-UE} \quad (7.2)$$

$$T_{UE-HA} \geq 2 \cdot T_{UE-SCSCF} + \frac{11 \cdot n - 6}{2} \cdot T_{UE-UE} + (n-1) \cdot 5 - 100 \quad (7.3)$$

Table 7.4 shows some examples of T[UE_A–HA] thresholds above which the MIP handover times become larger than the SIP handover times. These thresholds are derived from Inequation (7.3) for specific combinations of number of media streams, T[UE_A–S-CSCF] and T[UE_A–UE_B] values.

Figure 7.2 already showed that when the network delays are the same between UE_A and its HA and UE_A and its S-CSCF are equal, MIP-based mobility outperforms the SIP-based approach. Table 7.4 highlights how much longer T[UE_A–HA] has to be compared to T[UE_A–S-CSCF] so that SIP performs macro handovers faster than IMS. For example,

when the session has only one media stream and the one-way delays delays between the UEs are short (50ms) and T[UE_A–S-CSCF] is 100ms long, T[UE_A–HA] can be up to 2.25 times bigger than T[UE_A–S-CSCF] before it is not worth implementing MIP mobility in the IMS—this can also be seen on the first subplot of Figure 7.2.
Also, remember that the packet size has not been considered in the analytical model. In real systems, MIP messages are much smaller than SIP signaling messages cf. Tables 4.5 and 5.2 in [Fathi06] for detailed SIP and MIP message sizes), which would influence the results even more in favor of the MIP solution. Communication delays are especially sensitive on the air interface where the frame loss probability is much higher than in wired parts of the system. Another factor in relation to frame losses on the air interface is the number of messages that exchanged to and from both UEs, which is also much higher with the SIP solution.

Table 7.4, Examples of T[UE_A–HA] thresholds

# of streams	Inequation	T[UE–SCSCF]	T[UE–UE]	T[UE–HA] thresholds
1	$T_{UE-HA} \geq 2 \cdot T_{UE-SCSCF} + 2{,}5 \cdot T_{UE-UE} - 100$	100	50 100 150	225 350 475
2	$T_{UE-HA} \geq 2 \cdot T_{UE-SCSCF} + 8 \cdot T_{UE-UE} - 95$	100	50 100 150	505 905 1305
3	$T_{UE-HA} \geq 2 \cdot T_{UE-SCSCF} + 13{,}5 \cdot T_{UE-UE} - 90$	100	50 100 150	785 1460 2035

7.5. Conclusion

Our novel mobility solution is based on a new type of PDP context that can be activated quickly and without SIP. It offers the advantages of being simpler and more efficient: fewer messages are involved, especially on the air interface, which is prone to longer delays and more packet losses, and does not require end-to-end communications between UEs. The use of the authorization token given by the IMS in the old access network ensures service control in the new access network, while considerably shortening the handover times. Except in a very specific scenario, the MIP solution will perform much faster than the standard SIP procedures (up to 4.5 times faster in the best conditions).
The drawbacks of the solution are limited:

- small standardization and implementation efforts in comparison to the huge gains in terms of handover delays;
- little additional cost for the user until the signaling and data flows are separated again;
- short period with no SIP services until the UE can re-REGISTER with the IMS.

The solution proposed assumes that the QoS level granted in AN_{new} for the MSH PDP context matches those in AN_{old}. If the levels in AN_{old} and AN_{new} differ, UE_A needs to perform end-to-end QoS negotiation again after the handover. This QoS negotiation

would impact handover delays considerably so a fast mechanism needs to be designed in order to keep those delays reasonable. In case of proactive handover, the UE could perform the negotiation before leaving AN_{old}.
Future work could also investigate the benefits of using alternative MIPv6-based solutions (FMIP, HMIP, etc.) to better improve the handover delays.
Finally, since the IMS was designed to be network-independent, we need to analyze the portability of our solution to other access networks, such as WLAN, using ongoing research that focuses on integration of access control functionalities in such access networks.

8. Conclusions and Outlook

8.1. Summary

The IMS is becoming a key component in IP-based service provisioning. Because of the requirements steming from the end-users and the critical involvement of the control procedures in multimedia session management, the IMS should be dependable without getting slower or invasive for the access network it controls.
The contributions in this thesis focus on the dependability/performance tradeoff and investigate means to improve performance while maintaining the dependability levels guaranteed by the fault tolerance mechanisms in their 'standard' setup.
Two main fault scenarios were considered that require appropriate recovery strategies:

- IMS servers sometimes crash for some periods of time so they are replicated, allowing for failovers to mask the servers failures.
- Sometimes, communications with external entities are made impossible because of faults affecting single points of failure. The best option in this case is to attach to another access network.

In Chapter 2, the complex IMS procedures and their interactions with access network entities are explained, as well as the SIP mechanisms these procedures rely on.

In Chapter 3, two approaches to support server replication are introduced, their integration with the IMS platform is shortly discussed and a solution is proposed that allocates the recovery decisions to the P-CSCF instead of the service clients themselves.

In Chapter 4, the models used to evaluate the dependability/performance tradeoff in the standard and replicated IMS are thoroughly explained, and a set of output metrics suiting the models is defined. These metrics are

- Dependability, the successful transaction probability
- SAT, the average successful transaction completion time
- Load, the average load per transaction

A first analysis of the environment input variables is qualitatively discussed and supported by simulation results from the standard IMS scenario, motivating a restricted set of input variables for the main 'optimal fault tolerance configuration' analysis. It shows that the SIP traffic load is not very important and the little effects it has on the output metrics can be evaluated analytically. Packet losses are not likely to cause transaction failures thanks to retransmissions but they surely increase the service access time a lot and also the load, to a lesser extent. The effects of server failures on the system are quite different from those of packet losses: dependability can be greatly affected as the probability that servers are OFF rises while the service access time does not increase by much. Another important measure proved to be the ratio between the round trip time

and the average time to repair or, more precisely, the ratio between the maximum transaction lifetime and the time to repair. For the rare systems where server failures do not last more than a few seconds, the reaction of the three metrics to packet loss and server unavailibity are different because the round trip time/time to repair ratio becomes abnormally big; so, then, dependability still decreases as packet losses and server unavailability increase but service access time and load increase much faster than in smaller round trip time/time to repair scenarios.
Several fault tolerance parameters are considered in the second phase of the analysis, where the expected effects of the input parameters on the tradeoff are compared to the simulation results. In general, for the model settings considered, fault tolerance configurations with more failovers are more dependable and more performant than other configurations for which max_request is the same or even higher sometimes. The drawbacks of deploying many replicated servers are extra load (but not that much in comparison to the dependability and service access time gains) and additional deployment costs such the expenses for many servers and all the wiring and maintanance, etc. Having even more servers than necessary for the failover, or increasing the heartbeat rate both seem like promising tracks to investigate for improvement. Finally, when the distribution of network delays is known, it is possible to adapt the SIP timeout by fixing a threshold on the percentage of network delays that the timeout should be bigger than. This way, many retransmissions due to late responses from available servers could be avoided, which provides great best improvements in terms of dependability and maintain both are introduced service access time and the load almost at the standard IMS levels.
Methods that use the conclusions of the analysis to select the optimal fault tolerance configuration are suggested and an example illustrates the direct application of the results for the replicated IMS system.

In Chapter 5, an inconsistency definition that suits the IMS system is given. Then, a framework is presented that can be used to evaluate the metric previously defined. The evaluation approach relies on contributing factors that are influenced by the server selection policy, traffic model, state replication model, and network characteristics such as packet losses and network delays.
The inconsistency evaluation framework is verified by comparing (1) inconsistency values measured in an experimental SIP system and (2) inconsistency values generated with the framework, where the impacting factors are calculated with a mix of inputs from the system specifications and experimental tests. It is pretty accurate overall but seems to perform better with lower network delays.
The framework was primarily designed to help dynamic commitment protocols to determine precisely the delay they should wait before updating a state. The framework can also be used to help the architecture/protocol design of a system by determining the expected inconsistency levels. Accordingly, the state dissemination protocol that helps meet the predefined target inconsistency/performance tradeoff can be chosen.

In Chapter 6, several mobility solutions for macro handovers are presented. Then, it is argued that in the IMS environment MIP is best suited to support macro mobility. The addressing conflicts between MIP and the IMS are highlighted and a set of protocol/function extension is proposed to get around these conflicts. At this stage, the

MIP+IMS architecture provides session continuity, which the standard IMS does not, but the handover time is not improved.

Chapter 7 proposes a novel MIP-based macro mobility solution for IMS environments that shortens the handover times. To do so, secure media authorization information generated, and saved, at AN_{old} is transferred to AN_{new} so that a single, novel PDP context is created at AN_{new}. The advantage of using only one PDP context is that the UE can start exchanging data packets much quicker than with a standard SIP handover/session setup. Indeed, the handover times with this solution are independent from the number of ongoing data streams at AN_{old} before the handover, while the standard IMS solution is almost directly proportional to the number of ongoing sessions. Therefore, the new mobility design is especially improving the handover time of sessions with multiple data stream such as a Skype session where there would simultaneously be voice, video and instant messages.

8.2. Outlook

There are a number of open issues raised in the thesis that could be interesting for further research.

8.2.1. Optimal Fault Tolerance Configuration

The average service access time affects the distribution of SIP transaction initiation times. By implementing the SIP traffic model at each PU with the possibility to support multiple simultaneously pending transactions it would probably cancel the counter-intuitive results showing that increasing the heartbeat frequency or the pool size yields lower dependability.
This should be done in conjunction with fine tuning the server selection policy so that there is a good compromise between dependability and load: e.g. should the clients keep sending SIP requests when the PElist is empty (i.e. keep the server address in the list upon failover but indicate that the server was actually unavailable when contacted so it should be contacted again only after the list has been declared ‘empty’). If the clients do keep sending the SIP requests, chances are that dependability slightly increases at a higher load cost, in proportions to the dependability gains.

The model should include more realistic network delays and packet loss distributions. E.g. a SIP request sent by one client to a given server will experience network delays that are completely independent from the network delays experienced by another SIP request to the same server sent by a different client, which is not so realistic. Also the exponentially distributed network delays can have values close to ‘0’, which once again is not realistic.
Scenarios with more realistic round trip time/time to repair ratios should also be investigated. Unfortunately, this would imply much longer simulation times (48hrs per test…?).

Implementing load-dependent processing times would also make the model more realistic because clients implement a server selection policy that makes them all pick the same servers in the same order. Therefore, it could be possible that the servers become slower or even fully overloaded. The effects of server selection policy are interesting to investigate to broaden the holistic view on the inter-relation between dependability and performance even more.

Additional failure detection schemes could be tested that would introduce additional feedback from the clients to the name server and the SIP servers to the name server about SIP servers' status.

8.2.2. State Consistency

The validation exercise showed that the processing times of the prototype SIP server had a long-tail distribution. So in order to verify the evaluation framework in a more 'controlled' setup, the framework could be implemented in the SAN models and compared to simulation inconsistency results.

It would be interesting to develop concrete procedures for run time inconsistency evaluation and test it with Möbius to assess the capacity of the framework to fulfil the task it was originally designed for

8.2.3. MIP+IMS Macro Mobility

Experimental setups are always a good option when it comes to provide a proof-of-concept, e.g. [Larsen06b] proved that security associations can be moved from one P-CSCF to another. Here, the goal would be to test the feasibility/complexity of the interworking approach and find out more in detail the type of constraints that could prevent the implementation of the novel MSH PDP context.

In addition to service continuity (important to the user), charging continuity should also be considered in our scenario. Even though it greatly depends on operators' charging policies, we suggest that the session state logged at $P\text{-}CSCF_{old}$ for the ongoing session is transferred to $P\text{-}CSCF_{new}$ in order to create a new charging state at AN_{new}. More media-related information is provided at the IMS level than what is specified in the PDP context activation requests: the SDP content indicates the codecs used in the session. This allows appropriate charging based on the type(s) of media carried in the MSH PDP context.

Potential faults have not been taken into account when designing the fast MIP-based IMS macro handover but it would be interesting to extend the macro handover solution to consider fault scenarios. Faults would definitely increase the handover time. In more severe cases where, say, the MIP home agent is crashed, the handover to AN_{new} might not even be feasible at all. IETF has started a new standardization activity that focuses on MIP architectures with multiple home agents [Faizan04] so maybe solutions already exist that can be integrated with the mobility solution to make it dependable as well.

A. SIP Specifics

A.1. SIP Responses

Table A.1, SIP response codes and their meaning

Type	Code	Meaning
Information	100	Trying
	180	Ringing
	181	Call is being forward
	182	Queued
Success	200	OK
Redirection	300	Multiple choices
	301	Moved permanently
	302	Moved temporarily
	303	See other
	305	Use proxy
	380	Alternative service
Client error	400	Bad request
	401	Unauthorized
	402	Payment required
	403	Forbidden
	404	Not found
	405	Method not allowed
	406	Not acceptable
	407	Proxy authentication required
	408	Request timeout
	409	Conflict
	410	Gone
	411	Length Required
	413	Request Entity Too Large
	414	Request-URI Too Large
	415	Unsupported Media Type
	420	Bad Extension
	480	Temporarily not available
	481	Call Leg/Transaction Does Not Exist
	482	Loop Detected
	483	Too Many Hops

	484	Address Incomplete
	485	Ambiguous
	486	Busy Here
Server Error	500	Internal
	501	Not Implemented
	502	Bad Gateway
	503	Service Unavailable
	504	Gateway Time-out
	505	SIP Version not supported
Global Failure	600	Busy Everywhere
	603	Decline
	604	Does not exist anywhere
	606	Not Acceptable

A.2. SIP Headers

There are the different headers belonging to the four types of SIP headers.

A.2.1. General Headers

- Call ID: it is mandatory in all SIP messages. It permits to identify a call between two user agents.
- Contact: it is present in INVITE, ACK, REGISTER requests and 1xx, 2xx, 3xx responses. It provides the URL where the user can be called or reached.
- CSeq: Command Sequence is required in every request. It contains a decimal number that increases at each request.
- Date: it indicates the time when the request or response has been sent.
- Encryption: it is used to show the portion of the SIP message that has been encrypted. Encryption provides privacy for end users.
- From: it indicates the initiator URL.
- Organization: it indicates the organization to which the initiator belongs.
- Retry-After: it indicates when a resource will be available again.
- Subject: it indicates the subject of the session.
- Supported: it indicates the options supported by a user agent or server.
- Time stamp: it is used to mark the exact time when a request was generated.
- To: it indicates the recipient SIP URL.
- User Agent: it conveys information about the initiator of the message.
- Via: it records the SIP path taken by the request and this path is used to route a response back to the initiator.

A.2.2. Request Headers

- Accept: it indicates what type of media is acceptable.

- Accept - Encoding: the same function as Accept header, encoding is ensuring that a SIP message with a large message body fits inside a single UDP packet.
- Accept – Language: it specifies in which language the messages are written.
- Authorization: it is used by a user to authenticate itself in a sever.
- Hide: it indicates that the client wants the path mentioned in the VIA header field to be hidden from subsequent proxies and user agents.
- Max – Forwards: it indicates the maximum number of hops that the SIP request may take.
- Priority: it indicates the urgency of the request.
- Proxy – Authorization: it allows the client to identify itself to a proxy which requires authentication.
- Route: it indicates the route taken by the request.
- Proxy – Require: it lists the features required by the user agent or the proxy in order to process request.
- Record – Route: it imposes all requests to go through a specified proxy.
- Require: it lists all the features the UAC requires the UAS to support.
- Response Key: it can be used by a client to request the key that the called user agent uses to encrypt the response with.

A.2.3. Response Header

- Proxy–Authenticate: it is used in the 407 Proxy–Authentication Required response.
- Server: it contains information about the software used by the UAS to process request.
- Unsupported: it indicates the features not supported by the server.

A.2.4. Entity Headers

- Content–Encoding: it allows the UAS to know the decoding scheme to interpret the message body.
- Content–length: it indicates the number of bytes in the message body.
- Content–Type: it specifies the media type in the body message.
- Expires: it indicates the time within the request is valid.

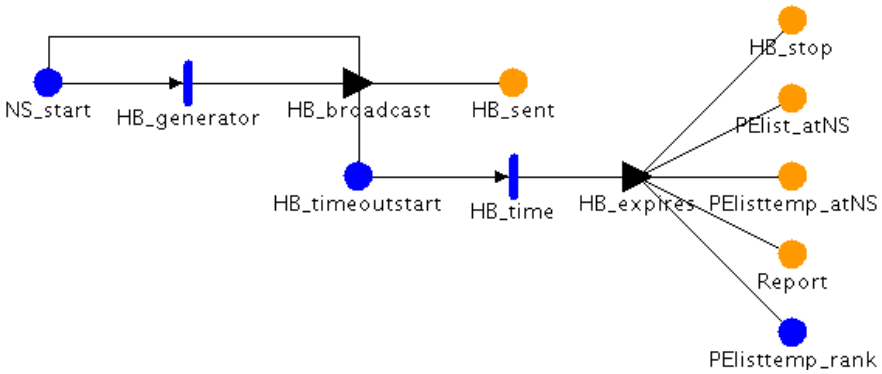
B. SAN/Möbius Models

This work was conducted in collaboration with Alessandro Daidone (University of Florence, Italy). Thanks for his insights on SAN modelling and Möbius implementation.

B.1. Atomic Models

Each atomic model of the composed model described in Section 4.3 is shown below.

B.1.1. NS Model



Place Names	Initial Markings
HB_sent	0
HB_stop	0
HB_timeoutstart	0
NS_start	1
PElist_atNS	Number_PE
PElisttemp_atNS	Number_PE
PElisttemp_rank	0
Report	0

Timed Activity:	HB_generator
Distribution Parameters	Value HB_intertime
Activation Predicate	(none)
Reactivation Predicate	(none)

Timed Activity:	HB_time
Distribution	Value

Parameters	HB_timeout
Activation Predicate	(none)
Reactivation Predicate	(none)

Output Gate:	HB_broadcast

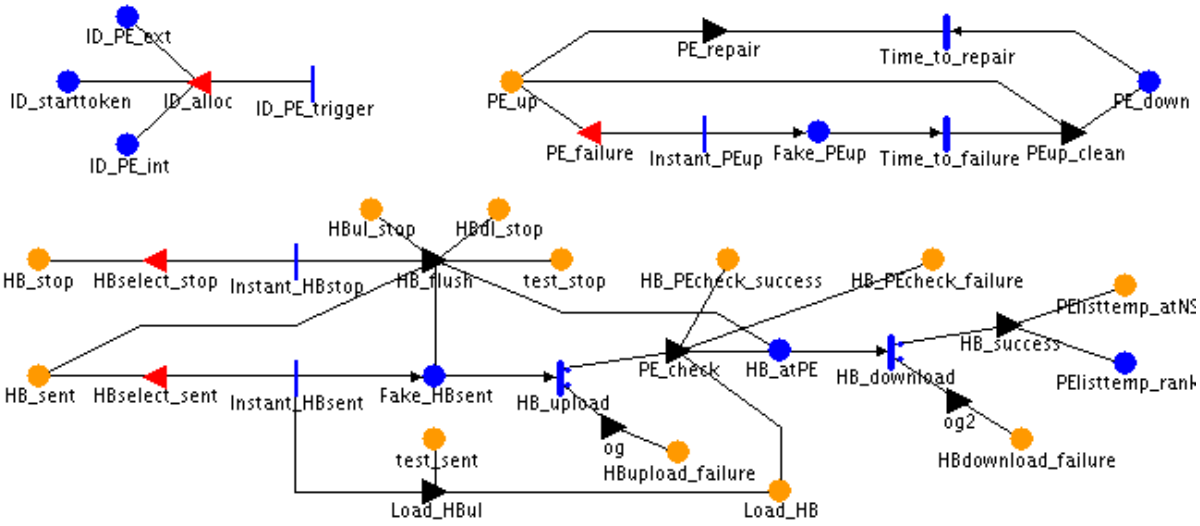
Function	`for (int i=0; i<Number_PE; i++)`
	`{`
	`  HB_sent->Index(i)->Mark() = 1;`
	`}`
	
	`HB_timeoutstart->Mark() ++;`
	
	`NS_start->Mark() = 1;`

Output Gate:	HB_expires

Function	`for (int i=0; i<Number_PE; i++)`
	`{`
	`  HB_stop->Index(i)->Mark() = 1;`
	`}`
	
	`for (int j=0; j<Number_PE; j++)`
	`{`
	`  PElist_atNS->Index(j)->Mark() = PElisttemp_atNS->Index(j)->Mark();`
	`}`
	
	`for (int h=0; h<Number_PE; h++)`
	`{`
	`  PElisttemp_atNS->Index(h)->Mark() = 0;`
	`}`
	
	`for (int k=0; k<Number_PU; k++)`
	`{`
	`  Report->Index(k)->Mark() = 1;`
	`}`
	
	`PElisttemp_rank->Mark() = 0;`

B.2. Atomic Models

B.2.1. PE Model



Place Names	Initial Markings
Fake_HBsent	0
Fake_PEup	0
HB_PEcheck_failure	0
HB_PEcheck_success	0
HB_atPE	0
HB_sent	0
HB_stop	0
HBdl_stop	0
HBdownload_failure	0
HBul_stop	0
HBupload_failure	0
ID_PE_ext	0
ID_PE_int	Number_PE
ID_starttoken	1
Load_HB	0
PE_down	0
PE_up	1
PElisttemp_atNS	Number_PE
PElisttemp_rank	0
test_sent	0
test_stop	0

Timed Activity:	HB_download
Distribution	Rate

Parameters	download_PE_NS
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	**case 1** 1-PER **case 2** PER
Timed Activity:	**HB_upload**
Distribution Parameters	**Rate** upload_NS_PE
Activation Predicate	(none)
Reactivation Predicate	(none)
Case Distributions	**case 1** 1-PER **case 2** PER
Timed Activity:	**Time_to_failure**
Distribution Parameters	**Rate** TTF
Activation Predicate	(none)
Reactivation Predicate	(none)
Timed Activity:	**Time_to_repair**
Distribution Parameters	**Rate** TTR
Activation Predicate	(none)
Reactivation Predicate	(none)

Instantaneous Activities Without Cases:
ID_PE_trigger
Instant_HBsent
Instant_HBstop
Instant_PEup

Input Gate:	**HBselect_sent**
Predicate	(ID_starttoken->Mark() == 0) && (HB_sent->Index(ID_PE_int->Mark())->Mark() == 1)

Function	//fprintf (stderr, "__INPUT function of HBselect_sent__\n");
	
	int time = LastActionTime;
	
	//fprintf (stderr, "Time = %i \n", time);

	//fprintf (stderr, "ID_PE_int = %i \n", ID_PE_int->Mark());
	//fprintf (stderr, "HB_sent = %i \n", HB_sent->Index(ID_PE_int->Mark())->Mark());
	//fprintf (stderr, "__INPUT function of HBselect_sent DONE__\n\n");
	
	HB_sent->Index(ID_PE_int->Mark())->Mark() = 0;

Input Gate:	HBselect_stop
Predicate	(ID_starttoken->Mark() == 0) && (HB_stop->Index(ID_PE_int->Mark())->Mark() == 1)
Function	HB_stop->Index(ID_PE_int->Mark())->Mark() = 0;

Input Gate:	ID_alloc
Predicate	(ID_starttoken->Mark() == 1)
Function	ID_PE_int->Mark() = ID_PE_ext->Mark(); ID_PE_ext->Mark() ++; ID_starttoken->Mark() = 0; fprintf (stdout, "____CIAO ID alloc done CIAO____ \n");

Input Gate:	PE_failure
Predicate	(ID_starttoken->Mark() == 0) && (PE_up->Index(ID_PE_int->Mark())->Mark() == 1) && (Fake_PEup->Mark() == 0)
Function	;

Output Gate:	HB_flush

Function	HB_sent->Index(ID_PE_int->Mark())->Mark() = 0;
	
	if (Fake_HBsent->Mark() == 1)
	{
	  Fake_HBsent->Mark() = 0;
	  HBul_stop->Mark() ++;
	}
	
	if (HB_atPE->Mark() == 1)
	{
	  HB_atPE->Mark() = 0;
	  HBdl_stop->Mark() ++;
	}
	
	test_stop->Mark() ++;

Output Gate:	HB_success

Function	`PElisttemp_atNS->Index(ID_PE_int->Mark())->Mark() = (Number_PE - PElisttemp_rank->Mark());`
	
	`PElisttemp_rank->Mark() ++;`

Output Gate:	**Load_HBul**
Function	`test_sent->Mark() ++;` `Load_HB->Mark() ++;`

Output Gate:	**PE_check**

Function	`if ((Fake_PEup->Mark() == 1) || (PE_down->Mark() == 0))`
	`{`
	`  HB_PEcheck_success->Mark() ++;`
	`  Load_HB->Mark() ++;`
	`  HB_atPE->Mark() ++;`
	`}`
	
	`else`
	`  HB_PEcheck_failure->Mark() ++;`

Output Gate:	**PE_repair**

Function	`PE_up->Index(ID_PE_int->Mark())->Mark() = 1;`
	
	`// for (int i=0; i<Number_PE; i++)`
	`// {`
	`//   (*trout)<< "PEindex_" << ID_PE_int->Mark() << " = " << PE_up->Index(ID_PE_int->Mark())->Mark() << endl;`
	`// }`
	
	`// (*trout) << endl;`

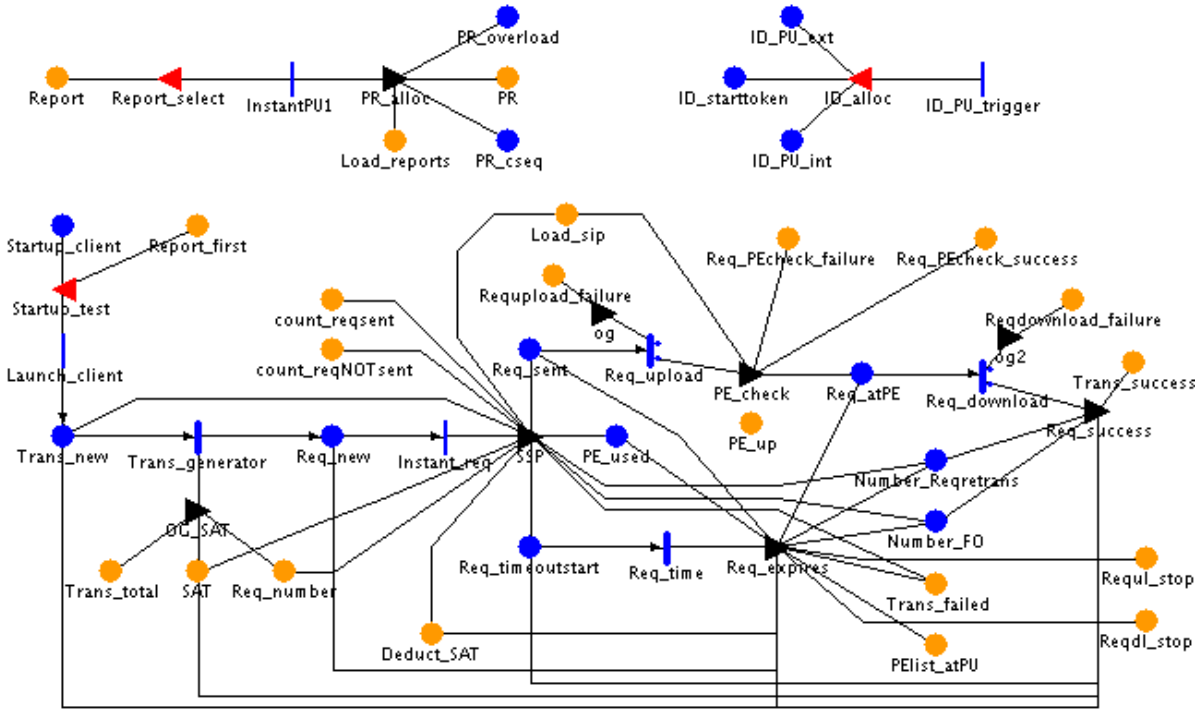
Output Gate:	**PEup_clean**

Function	`PE_up->Index(ID_PE_int->Mark())->Mark() = 0;`
	
	`PE_down->Mark() = 1;`

Output Gate:	**og**
Function	`HBupload_failure->Mark() ++;`

Output Gate:	**og2**
Function	`HBdownload_failure->Mark() ++;`

B.2.2. PU Model



Place Names	Initial Markings
Deduct_SAT	0
ID_PU_ext	0
ID_PU_int	Number_PU
ID_starttoken	1
Load_reports	0
Load_sip	0
Number_FO	0
Number_Reqretrans	0
PE_up	1
PE_used	Number_PE
PElist_atPU	Number_PE
PR	0
PR_cseq	1
PR_overload	0
Report	0
Report_first	0
Req_PEcheck_failure	0
Req_PEcheck_success	0
Req_atPE	0
Req_new	0

Req_number	0
Req_sent	0
Req_timeoutstart	0
Reqdl_stop	0
Reqdownload_failure	0
Requl_stop	0
Requpload_failure	0
SAT	0
Startup_client	1
Trans_failed	0
Trans_new	0
Trans_success	0
Trans_total	0
count_reqNOTsent	0
count_reqsent	0

Timed Activity:	**Req_download**
Distribution Parameters	**Rate** `download_PE_PU`
Activation Predicate	(none)
Reactivation Predicate	(none)

Case Distributions	**case 1**
	
	`PER`
	case 2
	
	`1-PER`

Timed Activity:	**Req_time**
Distribution Parameters	**Value** `Req_timeout*(pow(2,(Req_number->Mark()-1)))`
Activation Predicate	(none)
Reactivation Predicate	(none)

Timed Activity:	**Req_upload**
Distribution Parameters	**Rate** `upload_PU_PE`
Activation Predicate	(none)
Reactivation Predicate	(none)

Case Distributions	**case 1**
	
	`PER`
	case 2
	
	`1-PER`

Timed Activity:	**Trans_generator**
	Rate

Distribution Parameters	Trans_intertime
Activation Predicate	(none)
Reactivation Predicate	(none)
Instantaneous Activities Without Cases:	
ID_PU_trigger	
InstantPU1	
Instant_req	
Launch_client	
Input Gate:	**ID_alloc**
Predicate	`(ID_starttoken->Mark() == 1)`

Function	`ID_PU_int->Mark() = ID_PU_ext->Mark();`
	`ID_PU_ext->Mark() ++;`
	`ID_starttoken->Mark() = 0;`

Input Gate:	**Report_select**
Predicate	`(ID_starttoken->Mark() == 0) && (Report->Index(ID_PU_int->Mark())->Mark() == 1)`
Function	`Report->Index(ID_PU_int->Mark())->Mark() = 0;`
Input Gate:	**Startup_test**
Predicate	`(Report_first->Mark() > 0) && (Startup_client->Mark() > 0)`
Function	`Startup_client->Mark() = 0;`
Output Gate:	**OG_SAT**

Function	`SAT->Mark() ++;`
	`Trans_total->Mark() ++;`
	`Req_number->Mark() = 0;`

Output Gate:	**PE_check**

Function	`if (PE_up->Index(PE_used->Mark())->Mark() == 1)`
	`{`
	`  Req_atPE->Mark() ++;`
	`  Load_sip->Mark() ++;`
	`  Req_PEcheck_success->Mark() ++;`
	`}`
	
	`else`
	`  Req_PEcheck_failure->Mark() ++;`

Output Gate:	**PR_alloc**
Function	`int index_PR=0;`

	while (index_PR<Number_PR)
	{
	  if (PR->Index(index_PR)->Mark() == 0)
	  {
	  PR->Index(index_PR)->Mark() = PR_cseq->Mark();
	  PR_cseq->Mark() ++;
	  Load_reports->Mark() ++;
	  break;
	  }
	  else
	  index_PR++;
	
	  if (index_PR == Number_PR) PR_overload->Mark() ++;
	}

Output Gate:	**Req_expires**

Function	if (Req_sent->Mark() > 0)
	{
	  Req_sent->Mark() = 0;
	  Requl_stop->Mark() ++;
	}
	
	if (Req_atPE->Mark() > 0)
	{
	  Req_atPE->Mark() = 0;
	  Reqdl_stop->Mark() ++;
	}
	
	if (Number_Reqretrans->Mark() == Max_Reqretrans)
	{
	  Number_Reqretrans->Mark() = 0;
	  PElist_atPU->Index(PE_used->Mark())->Mark() =
	Number_PE+1;
	  PE_used->Mark() = Number_PE;
	
	  if (Number_FO->Mark() == Max_FO)
	  {
	  Number_FO->Mark() = 0;
	  Deduct_SAT->Mark() += Req_number->Mark();
	  Req_number->Mark() = 0;
	  Trans_failed->Mark() ++;
	  SAT->Mark() --;
	  Trans_new->Mark() ++;
	  }
	  else
	  {
	  Number_FO->Mark() ++;
	  Req_new->Mark() ++;
	  }
	}
	
	else
	{

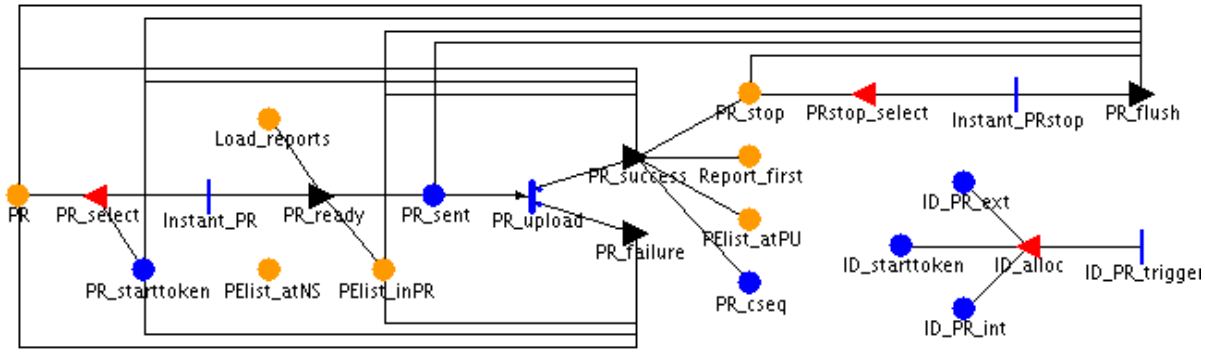
	Number_Reqretrans->Mark() ++; Req_new->Mark() ++; }
Output Gate:	**Req_success**
Function	Req_timeoutstart->Mark() = 0; SAT->Mark() --; Trans_success->Mark() ++; Trans_new->Mark() ++; // WE CHOSE TO RESET PE_RETRANS VALUES AFTER EVERY SUCCESSFUL TRANSACTION... Number_Reqretrans->Mark() = 0; Number_FO->Mark() = 0; Req_number->Mark() = 0;
Output Gate:	**SSP**

Function	// we chose to keep using the same PE as for the previous transaction if it is still 'up' (from the PU perspective)
	
	if ((PE_used->Mark() == Number_PE) || ((PE_used->Mark() < Number_PE) && (PElist_atPU->Index(PE_used->Mark())->Mark() >= Number_PE)))
	{
	  PE_used->Mark() = Number_PE;
	
	  int rankSSP = 0;
	  while (rankSSP < Number_PE+1)
	  // in this case, SSP also checks for PEs that are apparently OFF in the PElist_atPU (i.e. rank=Number_PE) BUT...
	  // ...SSP excludes PEs that were tried by the PU but could NOT be reached within Max_Reqretrans attempts (i.e. PE for which rank=Number_PE+1), see code Req_expires!!
	  {
	  int index=0;
	  while (index < Number_PE)
	  {
	  if (PElist_atPU->Index(index)->Mark() == rankSSP)
	  {
	  PE_used->Mark() = index;
	  break;
	  }
	  else
	  index++;
	  }
	
	  if (PE_used->Mark() == Number_PE)
	  rankSSP++;
	  else
	  break;
	  }
	
	  if (PE_used->Mark() < Number_PE)

	{
	Req_sent->Mark() ++;
	count_reqsent->Mark() ++;
	Load_sip->Mark() ++;
	Req_number->Mark() ++;
	Req_timeoutstart->Mark() ++;
	}
	else
	{
	count_reqNOTsent->Mark() ++;
	Trans_failed->Mark() ++;
	SAT->Mark() --;
	Deduct_SAT->Mark() += Req_number->Mark();
	Number_Reqretrans->Mark() = 0;
	Number_FO->Mark() = 0;
	Req_number->Mark() = 0;
	Trans_new->Mark() ++;
	}
	}
	
	else
	{
	Req_sent->Mark() ++;
	count_reqsent->Mark() ++;
	Load_sip->Mark() ++;
	Req_number->Mark() ++;
	Req_timeoutstart->Mark() ++;
	}

Output Gate:	**og**
Function	Requpload_failure->Mark() ++;
Output Gate:	**og2**
Function	Reqdownload_failure->Mark() ++;

B.2.3. PR Model



Place Names	Initial Markings
ID_PR_ext	0
ID_PR_int	Number_PR
ID_starttoken	1
Load_reports	0
PElist_atNS	Number_PE
PElist_atPU	Number_PE
PElist_inPR	Number_PE
PR	0
PR_cseq	1
PR_sent	0
PR_starttoken	1
PR_stop	0
Report_first	0
Timed Activity:	**PR_upload**
Distribution Parameters	**Rate** `upload_NS_PU`
Activation Predicate	(none)
Reactivation Predicate	(none)

Case Distributions	**case 1**
	`1-PER`
	case 2
	`PER`

Instantaneous Activities Without Cases:	
ID_PR_trigger	
Instant_PR	
Instant_PRstop	

Input Gate:	ID_alloc
Predicate	(ID_starttoken->Mark() == 1)
Function	ID_PR_int->Mark() = ID_PR_ext->Mark(); ID_PR_ext->Mark() ++; ID_starttoken->Mark() = 0;

Input Gate:	PR_select
Predicate	(ID_starttoken->Mark() == 0) && (PR_starttoken->Mark() == 1) && (PR->Index(ID_PR_int->Mark())->Mark() > 0)
Function	PR_starttoken->Mark() = 0;

Input Gate:	PRstop_select
Predicate	(ID_starttoken->Mark() == 0) && (PR_stop->Index(ID_PR_int->Mark())->Mark() == 1)
Function	PR_stop->Index(ID_PR_int->Mark())->Mark() = 0;

Output Gate:	PR_failure

Function	for (int i=0; i<Number_PE; i++)
	{
	  PElist_inPR->Index(i)->Mark() = 0;
	}
	
	PR->Index(ID_PR_int->Mark())->Mark() = 0;
	PR_starttoken->Mark() = 1;

Output Gate:	PR_flush
Function	PR_sent->Mark() = 0; PR_starttoken->Mark() = 1; PR->Index(ID_PR_int->Mark())->Mark() = 0; PR_stop->Index(ID_PR_int->Mark())->Mark() = 0; for (int i=0; i<Number_PE; i++) PElist_inPR->Index(i)->Mark() = 0;

Output Gate:	PR_ready

Function	for (int j=0; j<Number_PE; j++)
	{
	   PElist_inPR->Index(j)->Mark() = PElist_atNS->Index(j)->Mark();
	}
	
	Load_reports->Mark() ++;
	PR_sent->Mark() = 1;

Output Gate:	PR_success

Function	Report_first->Mark() ++;
	
	for (int i=0; i<Number_PE; i++)
	{
	  PElist_atPU->Index(i)->Mark() = PElist_inPR->Index(i)->Mark();
	  PElist_inPR->Index(i)->Mark() = 0;
	}
	
	if ((PR->Index(ID_PR_int->Mark())->Mark() +1) == PR_cseq->Mark())
	  PR_cseq->Mark() = 1;
	
	(*trout) << "PR_id_int=" << ID_PR_int->Mark() << endl;
	
	 for (int j=0; j<Number_PR; j++)
	  {
	  if ((0 < PR->Index(j)->Mark()) && (PR->Index(j)->Mark() < PR->Index(ID_PR_int->Mark())->Mark()))
	  PR_stop->Index(j)->Mark() = 1;
	  (*trout) << "PRstop(" << j << ")=" << PR_stop->Index(j)->Mark() << endl;
	  }
	
	(*trout) <<endl;
	
	PR->Index(ID_PR_int->Mark())->Mark() = 0;
	PR_starttoken->Mark() = 1;

B.3. Reward Model – Performance Variables

Many performance variables were created in order to verify that all the functions modeled behave as expected.

Performance Variable : avg__PE_ON		
Affecting Models	PE_maSSP	
Impulse Functions		
Reward Function	*(Reward is over all Available Models)* `return ((1.0*PE_maSSP->Fake_PEup->Mark())/Number_PE);`	
Simulator Statistics	Type	Time Averaged Interval of Time

Performance Variable : avg__PE_OFF		
Affecting Models	PE_maSSP	
Impulse Functions		
Reward Function	*(Reward is over all Available Models)* `return (1.0*(PE_maSSP->PE_down->Mark())/Number_PE);`	
Simulator Statistics	Type	Time Averaged Interval of Time

Performance Variable : Frequency_PE_ONOFFcycles		
Affecting Models	PE_maSSP	
Impulse Functions	PE->Time_to_failure	
	(Reward is over all Available Models) `return (1.0/(1.0*Number_PE));`	
	PE_maSSP->Time_to_repair	
	(Reward is over all Available Models) `return (1.0/(1.0*Number_PE));`	
Reward Function	*(Reward is over all Available Models)*	
Simulator Statistics	Type	Interval of Time

Performance Variable : HB_generated		
Affecting Models	NS_reg	
Impulse Functions	NS->HB_generator	
	(Reward is over all Available Models) `return 1.0;`	
	NS_reg->HB_generator	
	(Reward is over all Available Models) `return 1.0;`	
Reward Function	*(Reward is over all Available Models)*	
Simulator Statistics	Type	Interval of Time

Performance Variable : HB_sent_pl		
Affecting Models	PE_maSSP	

Impulse Functions		
Reward Function	*(Reward is over all Available Models)* `return ((1.0*PE_maSSP->test_sent->Mark())/Number_PE);`	
Simulator Statistics	Type	Instant of Time

Performance Variable : HB_stop_pl		
Affecting Models	PE_maSSP	
Impulse Functions		
Reward Function	*(Reward is over all Available Models)* `return ((1.0*PE_maSSP->test_stop->Mark())/Number_PE);`	
Simulator Statistics	Type	Instant of Time

Performance Variable : HBul_success		
Affecting Models	PE_maSSP	
Impulse Functions	PE->HB_upload_case1	
	(Reward is over all Available Models) `return 1.0;`	
	PE_maSSP->HB_upload_case1	
	(Reward is over all Available Models) `return 1.0;`	
Reward Function	*(Reward is over all Available Models)*	
Simulator Statistics	Type	Interval of Time

Performance Variable : HBul_failures_pl		
Affecting Models	PE_maSSP	
Impulse Functions		
Reward Function	*(Reward is over all Available Models)* `return ((1.0*PE_maSSP->HBupload_failure->Mark())/Number_PE);`	
Simulator Statistics	Type	Instant of Time

Performance Variable : HBul_stop_pl		
Affecting Models	PE_maSSP	
Impulse Functions		
Reward Function	*(Reward is over all Available Models)* `return ((1.0*PE_maSSP->HBul_stop->Mark())/Number_PE);`	
Simulator Statistics	Type	Instant of Time

Performance Variable : HB_PEcheck_success_pl		
Affecting Models	PE_maSSP	
Impulse Functions		
Reward Function	*(Reward is over all Available Models)* `return ((1.0*PE_maSSP->HB_PEcheck_success->Mark())/Number_PE);`	
Simulator Statistics	Type	Instant of Time

Performance Variable : HB_PEcheck_failures_pl	
Affecting Models	PE_maSSP

Impulse Functions		
Reward Function	*(Reward is over all Available Models)* `return ((1.0*PE_maSSP->HB_PEcheck_failure->Mark())/Number_PE);`	
Simulator Statistics	Type	Instant of Time

Performance Variable : HBdl_success		
Affecting Models	PE_maSSP	
Impulse Functions	PE->HB_download_case1	
	(Reward is over all Available Models) `return 1.0;`	
	PE_maSSP->HB_download_case1	
	(Reward is over all Available Models) `return 1.0;`	
Reward Function	*(Reward is over all Available Models)*	
Simulator Statistics	Type	Interval of Time

Performance Variable : HBdl_failures_pl		
Affecting Models	PE_maSSP	
Impulse Functions		
Reward Function	*(Reward is over all Available Models)* `return ((1.0*PE_maSSP->HBdownload_failure->Mark())/Number_PE);`	
Simulator Statistics	Type	Instant of Time

Performance Variable : HBdl_stop_pl		
Affecting Models	PE_maSSP	
Impulse Functions		
Reward Function	*(Reward is over all Available Models)* `return ((1.0*PE_maSSP->HBdl_stop->Mark())/Number_PE);`	
Simulator Statistics	Type	Instant of Time

Performance Variable : PR_generated_atNS		
Affecting Models	NS_reg	
Impulse Functions	NS->HB_time	
	(Reward is over all Available Models) `return 1.0;`	
	NS_reg->HB_time	
	(Reward is over all Available Models) `return 1.0;`	
Reward Function	*(Reward is over all Available Models)*	
Simulator Statistics	Type	Interval of Time

Performance Variable : PR_sent_byPU	
Affecting Models	SIPclient_reg_expTO
Impulse Functions	SIPclient->InstantPU1
	(Reward is over all Available Models)

	`return 1.0;`	
	SIPclient_reg_expreqTO->InstantPU1	
	(Reward is over all Available Models) `return 1.0;`	
	SIPclient_reg_expTO->InstantPU1	
	(Reward is over all Available Models) `return 1.0;`	
Reward Function	*(Reward is over all Available Models)*	
Simulator Statistics	Type	Interval of Time

Performance Variable : PR_overload_atPU_pl		
Affecting Models	SIPclient_reg_expTO	
Impulse Functions		
Reward Function	*(Reward is over all Available Models)* `return ((1.0*SIPclient_reg_expTO->PR_overload->Mark())/Number_PU);`	
Simulator Statistics	Type	Instant of Time

Performance Variable : PR_allowed_atPR		
Affecting Models	PR_reg	
Impulse Functions	Pending_report->Instant_PR	
	(Reward is over all Available Models) `return 1.0;`	
	PR_reg->Instant_PR	
	(Reward is over all Available Models) `return 1.0;`	
Reward Function	*(Reward is over all Available Models)*	
Simulator Statistics	Type	Interval of Time

Performance Variable : PR_success_atPR		
Affecting Models	PR_reg	
Impulse Functions	Pending_report->PR_upload_case1	
	(Reward is over all Available Models) `return 1.0;`	
	PR_reg->PR_upload_case1	
	(Reward is over all Available Models) `return 1.0;`	
Reward Function	*(Reward is over all Available Models)*	
Simulator Statistics	Type	Interval of Time

Performance Variable : PR_failures_atPR	
Affecting Models	PR_reg
Impulse Functions	Pending_report->PR_upload_case2

	(Reward is over all Available Models) `return 1.0;`	
	PR_reg->PR_upload_case2	
	(Reward is over all Available Models) `return 1.0;`	
Reward Function	*(Reward is over all Available Models)*	
Simulator Statistics	Type	Interval of Time

Performance Variable : PR_stop_atPR		
Affecting Models	PR_reg	
Impulse Functions	Pending_report->Instant_PRstop	
	(Reward is over all Available Models) `return 1.0;`	
	PR_reg->Instant_PRstop	
	(Reward is over all Available Models) `return 1.0;`	
Reward Function	*(Reward is over all Available Models)*	
Simulator Statistics	Type	Interval of Time

Performance Variable : SAT_pl		
Affecting Models	SIPclient_reg_expTO	
Impulse Functions		
Reward Function	*(Reward is over all Available Models)* `return ((1.0*SIPclient_reg_expTO->SAT->Mark())/Number_PU);`	
Simulator Statistics	Type	Interval of Time

Performance Variable : SAT_deduct_pl		
Affecting Models	SIPclient_reg_expTO	
Impulse Functions		
Reward Function	*(Reward is over all Available Models)* `return ((1.0*SIPclient_reg_expTO->Deduct_SAT->Mark())/Number_PU);`	
Simulator Statistics	Type	Instant of Time

Performance Variable : Req_started	
Affecting Models	SIPclient_reg_expTO
Impulse Functions	SIPclient->Instant_req
	(Reward is over all Available Models) `return 1.0;`
	SIPclient_reg_expreqTO->Instant_req
	(Reward is over all Available Models) `return 1.0;`
	SIPclient_reg_expTO->Instant_req
	(Reward is over all Available Models)

	`return 1.0;`	
Reward Function	*(Reward is over all Available Models)*	
Simulator Statistics	Type	Interval of Time

Performance Variable : Req_sent_pl		
Affecting Models	SIPclient_reg_expTO	
Impulse Functions		
Reward Function	*(Reward is over all Available Models)* `return ((1.0*SIPclient_reg_expTO->count_reqsent->Mark())/Number_PU);`	
Simulator Statistics	Type	Instant of Time

Performance Variable : Req_NOTsent_pl		
Affecting Models	SIPclient_reg_expTO	
Impulse Functions		
Reward Function	*(Reward is over all Available Models)* `return ((1.0*SIPclient_reg_expTO->count_reqNOTsent->Mark())/Number_PU);`	
Simulator Statistics	Type	Instant of Time

Performance Variable : Requl_success		
Affecting Models	SIPclient_reg_expTO	
Impulse Functions	SIPclient->Req_upload_case2	
	(Reward is over all Available Models) `return 1.0;`	
	SIPclient_reg_expreqTO->Req_upload_case2	
	(Reward is over all Available Models) `return 1.0;`	
	SIPclient_reg_expTO->Req_upload_case2	
	(Reward is over all Available Models) `return 1.0;`	
Reward Function	*(Reward is over all Available Models)*	
Simulator Statistics	Type	Interval of Time

Performance Variable : Requl_failures_pl		
Affecting Models	SIPclient_reg_expTO	
Impulse Functions		
Reward Function	*(Reward is over all Available Models)* `return ((1.0*SIPclient_reg_expTO->Requpload_failure->Mark())/Number_PU);`	
Simulator Statistics	Type	Instant of Time

Performance Variable : Requl_stop_pl		
Affecting Models	SIPclient_reg_expTO	
Impulse Functions		
Reward Function	*(Reward is over all Available Models)* `return ((1.0*SIPclient_reg_expTO->Requl_stop->Mark())/Number_PU);`	

Simulator Statistics	Type	Instant of Time

Performance Variable : Req_PEcheck_success_pl		
Affecting Models	SIPclient_reg_expTO	
Impulse Functions		
Reward Function	*(Reward is over all Available Models)* `return ((1.0*SIPclient_reg_expTO->Req_PEcheck_success->Mark())/Number_PU);`	
Simulator Statistics	Type	Instant of Time

Performance Variable : Req_PEcheck_failures_pl		
Affecting Models	SIPclient_reg_expTO	
Impulse Functions		
Reward Function	*(Reward is over all Available Models)* `return ((1.0*SIPclient_reg_expTO->Req_PEcheck_failure->Mark())/Number_PU);`	
Simulator Statistics	Type	Instant of Time

Performance Variable : Reqdl_success		
Affecting Models	SIPclient_reg_expTO	
Impulse Functions	SIPclient->Req_download_case2	
	(Reward is over all Available Models) `return 1.0;`	
	SIPclient_reg_expreqTO->Req_download_case2	
	(Reward is over all Available Models) `return 1.0;`	
	SIPclient_reg_expTO->Req_download_case2	
	(Reward is over all Available Models) `return 1.0;`	
Reward Function	*(Reward is over all Available Models)*	
Simulator Statistics	Type	Interval of Time

Performance Variable : Reqdl_failures_pl		
Affecting Models	SIPclient_reg_expTO	
Impulse Functions		
Reward Function	*(Reward is over all Available Models)* `return ((1.0*SIPclient_reg_expTO->Reqdownload_failure->Mark())/Number_PU);`	
Simulator Statistics	Type	Instant of Time

Performance Variable : Reqdl_stop_pl		
Affecting Models	SIPclient_reg_expTO	
Impulse Functions		
Reward Function	*(Reward is over all Available Models)* `return ((1.0*SIPclient_reg_expTO->Reqdl_stop->Mark())/Number_PU);`	
Simulator Statistics	Type	Instant of Time

Performance Variable : Trans_total_pl

Affecting Models	SIPclient_reg_expTO	
Impulse Functions		
Reward Function	*(Reward is over all Available Models)* `return ((1.0*SIPclient_reg_expTO->Trans_total->Mark())/Number_PU);`	
Simulator Statistics	Type	Instant of Time

Performance Variable : Trans_success_pl		
Affecting Models	SIPclient_reg_expTO	
Impulse Functions		
Reward Function	*(Reward is over all Available Models)* `return ((1.0*SIPclient_reg_expTO->Trans_success->Mark())/Number_PU);`	
Simulator Statistics	Type	Instant of Time

Performance Variable : Trans_failed_pl		
Affecting Models	SIPclient_reg_expTO	
Impulse Functions		
Reward Function	*(Reward is over all Available Models)* `return ((1.0*SIPclient_reg_expTO->Trans_failed->Mark())/Number_PU);`	
Simulator Statistics	Type	Instant of Time

Performance Variable : load_HB_pl		
Affecting Models	PE_maSSP	
Impulse Functions		
Reward Function	*(Reward is over all Available Models)* `return ((1.0*PE_maSSP->Load_HB->Mark())/Number_PE);`	
Simulator Statistics	Type	Instant of Time

Performance Variable : load_reports_pl		
Affecting Models	SIPclient_reg_expTO	
Impulse Functions		
Reward Function	*(Reward is over all Available Models)* `return ((1.0*SIPclient_reg_expTO->Load_reports->Mark())/Number_PU);`	
Simulator Statistics	Type	Instant of Time

Performance Variable : load_sip_pl		
Affecting Models	SIPclient_reg_expTO	
Impulse Functions		
Reward Function	*(Reward is over all Available Models)* `return ((1.0*SIPclient_reg_expTO->Load_sip->Mark())/Number_PU);`	
Simulator Statistics	Type	Instant of Time

B.4. Simulation Results

B.4.1. Output Metrics Calculation

The three output metrics were calculated from specific Möbius performance variables. Here is how they were calculated:

Dependability

```
Dep. = 100*Trans_success_pl/(Trans_success_pl+Trans_failed_pl)
```

SAT (successful transactions only)

```
SAT_per_transaction = 1000*(SAT_pl - SAT_deduct_pl)/Trans_success_pl
```

Dependability

```
Load_total = ((load_HB_pl*sizeHB)+(load_reports_pl*sizeREPORT)+ ...
           ...(load_sip_pl_10*sizeSIP)) /Trans_total
```

B.4.2. Result Graphs

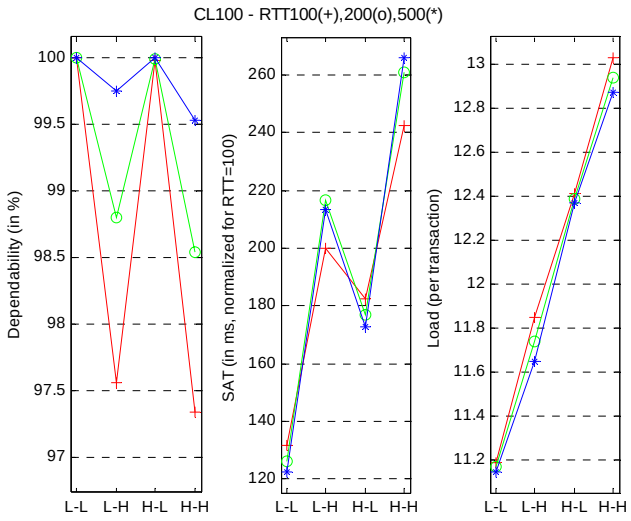


Fig.B.1, Standard SIP, CL=100s, comparing RTT

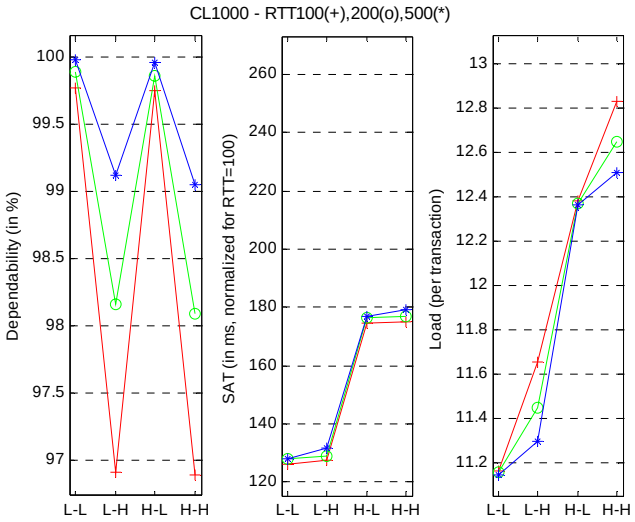


Fig.B.2, Standard SIP, CL=1000s, comparing RTT

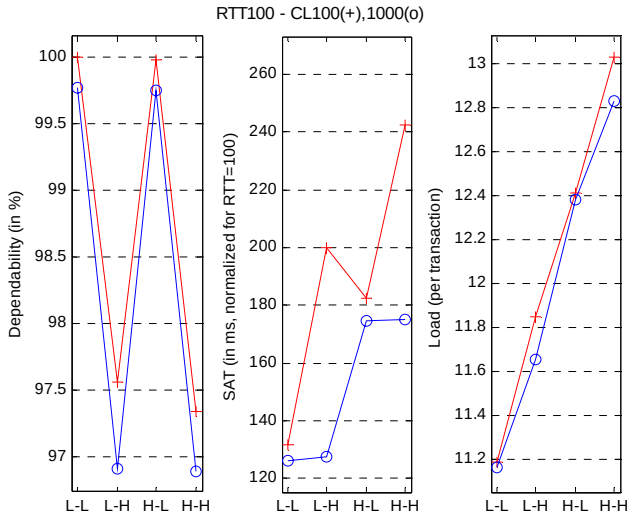


Fig.B.3, Standard SIP, RTT=100ms, comparing CL

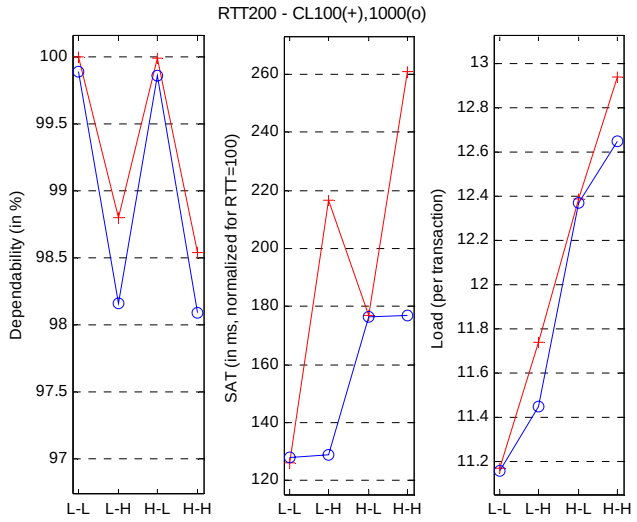


Fig.B.4, Standard SIP, RTT=200ms, comparing CL

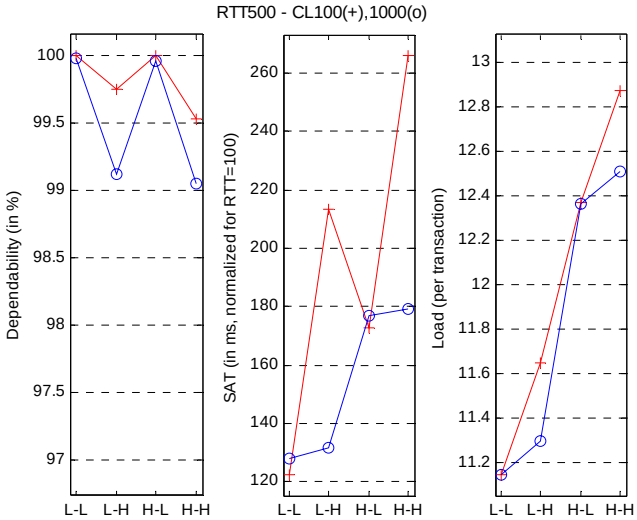


Fig.B.5, Standard SIP, RTT=500ms, comparing CL

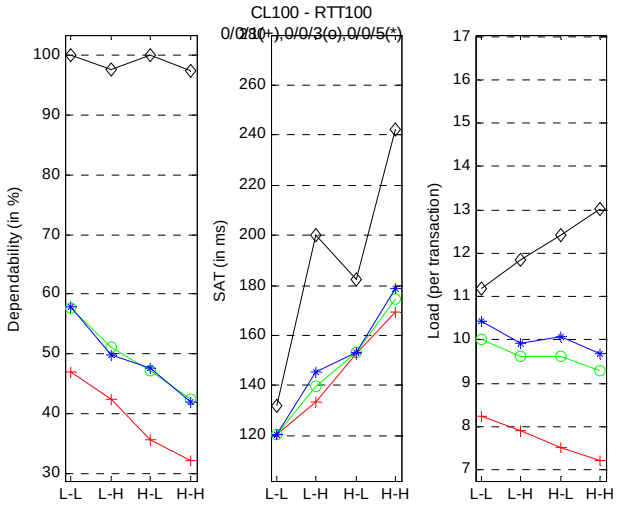


Fig.B.6, Regular, CL = 100s, RTT=100ms, FO=0, Retrans=0, comparing pool size

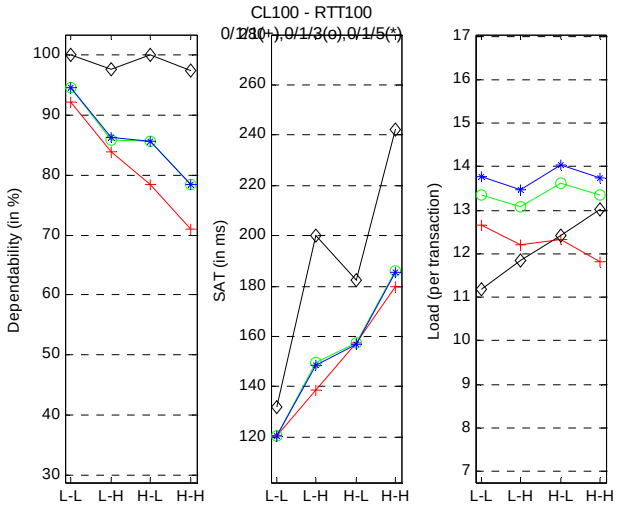


Fig.B.7, Regular, CL = 100s, RTT=100ms, FO=0, Retrans=1, comparing pool size

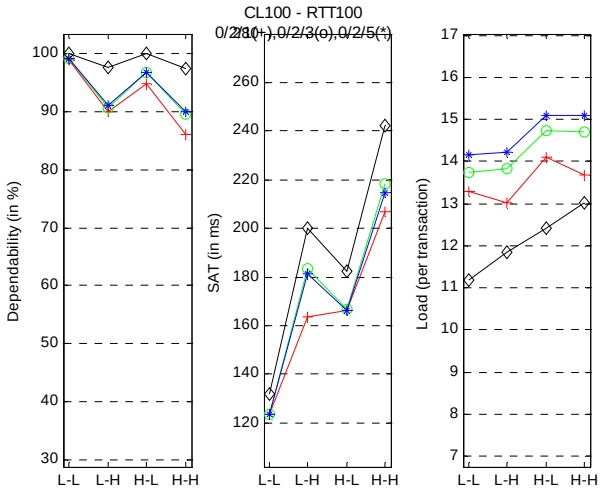


Fig.B.8, Regular, CL = 100s, RTT=100ms, FO=0, Retrans=2, comparing pool size

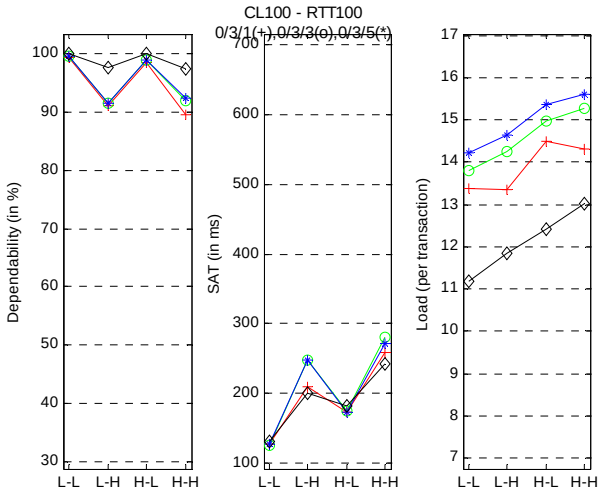


Fig.B.9, Regular, CL = 100s, RTT=100ms, FO=0, Retrans=3, comparing pool size

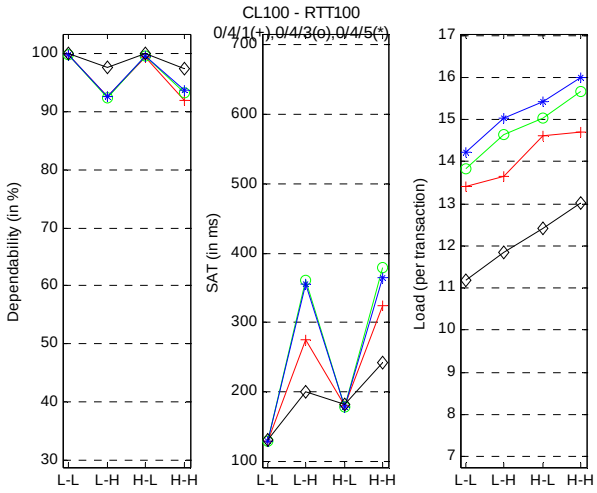


Fig.B.10, Regular, CL = 100s, RTT=100ms, FO=0, Retrans=4, comparing pool size

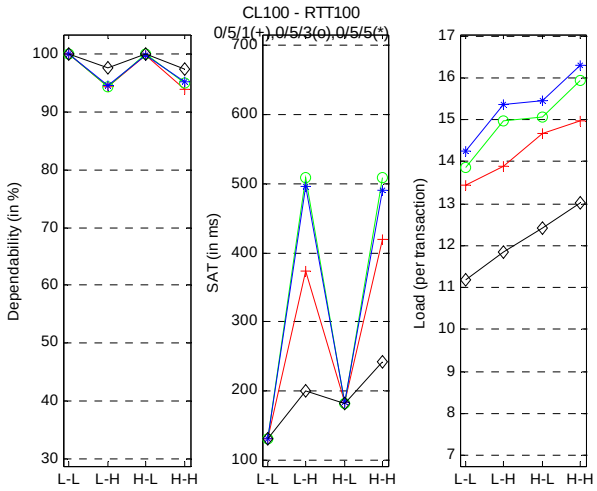


Fig.B.11, Regular, CL = 100s, RTT=100ms, FO=0, Retrans=5, comparing pool size

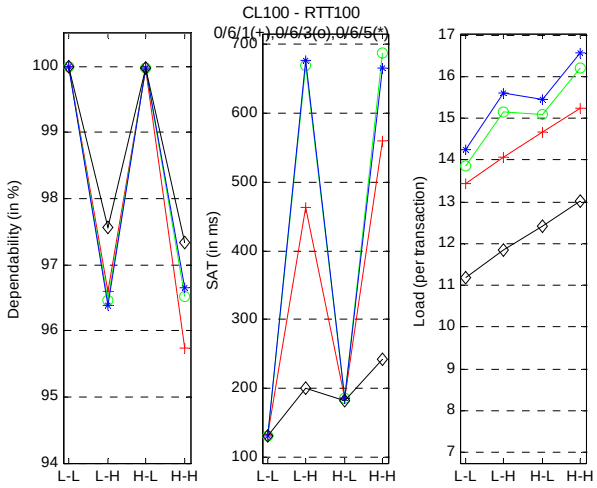


Fig.B.12, Regular, CL = 100s, RTT=100ms, FO=0, Retrans=6, comparing pool size

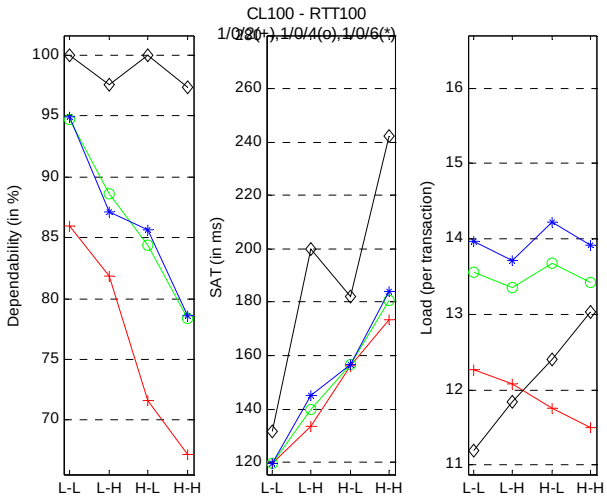


Fig.B.13, Regular, CL = 100s, RTT=100ms, FO=1, Retrans=0, comparing pool size

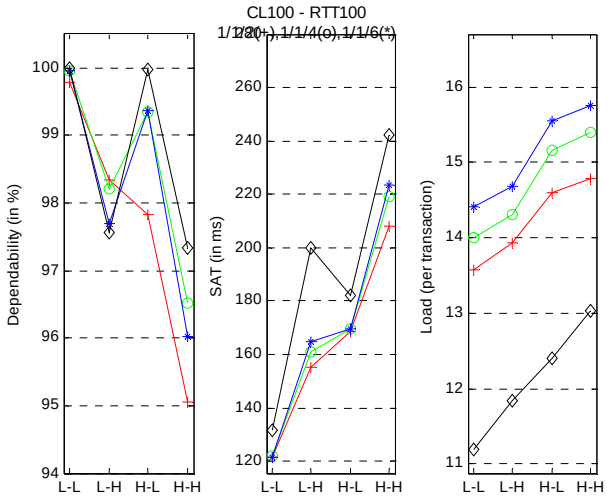


Fig.B.14, Regular, CL = 100s, RTT=100ms, FO=1, Retrans=1, comparing pool size

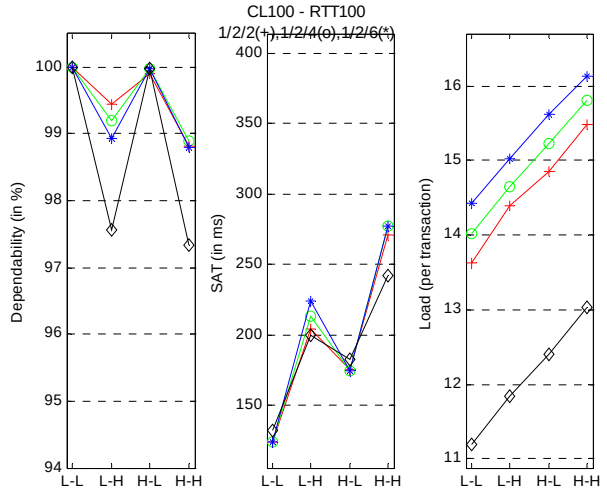


Fig.B.15, Regular, CL = 100s, RTT=100ms, FO=1, Retrans=2, comparing pool size

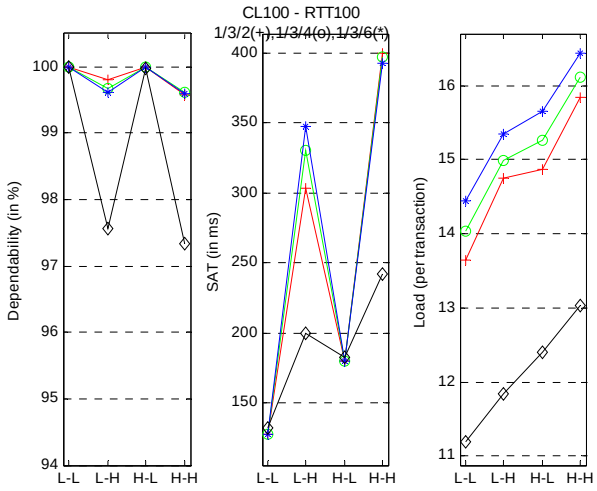


Fig.B.16, Regular, CL = 100s, RTT=100ms, FO=1, Retrans=3, comparing pool size

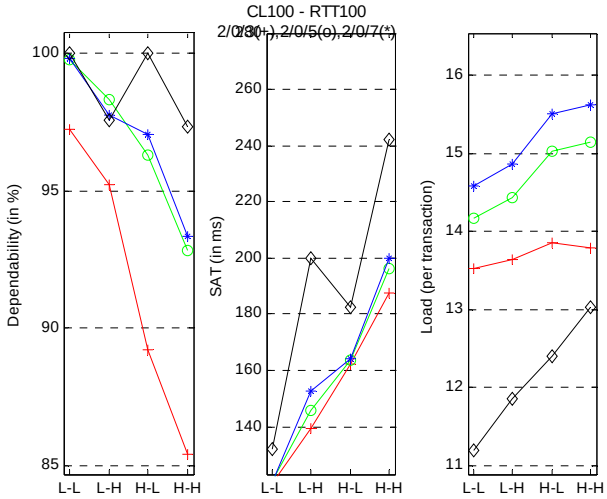


Fig.B.17, Regular, CL = 100s, RTT=100ms, FO=2, Retrans=0, comparing pool size

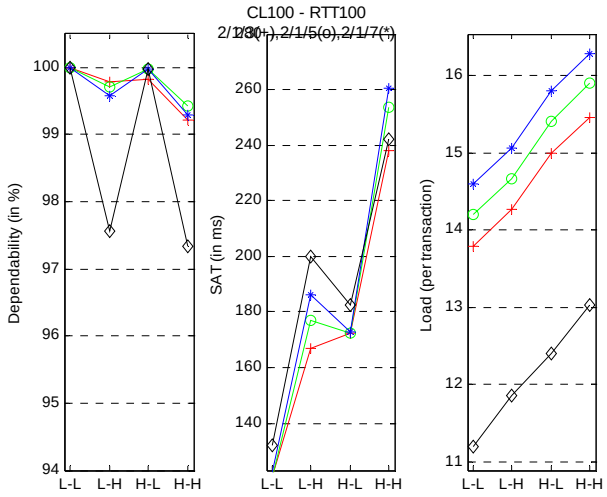


Fig.B.18. Regular, CL = 100s, RTT=100ms, FO=2, Retrans=1, comparing pool size

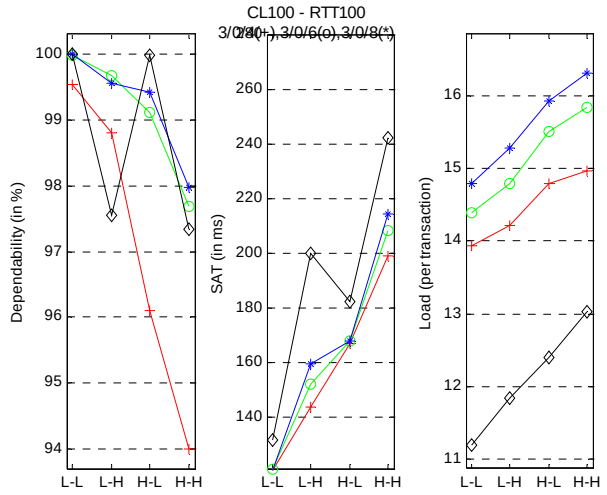


Fig.B.19, Regular, CL = 100s, RTT=100ms, FO=3, Retrans=0, comparing pool size

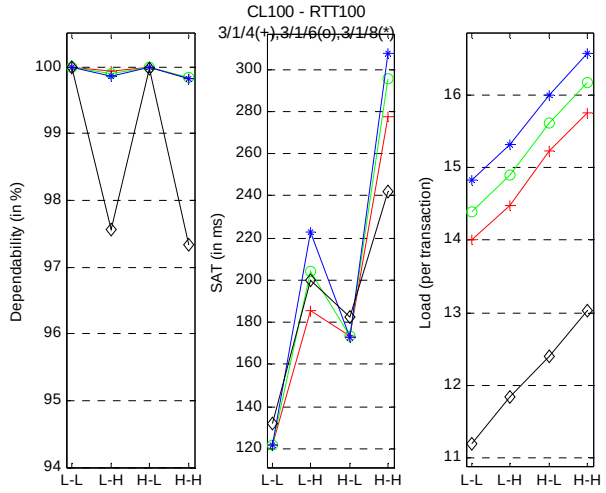


Fig.B.20, Regular, CL = 100s, RTT=100ms, FO=3, Retrans=1, comparing pool size

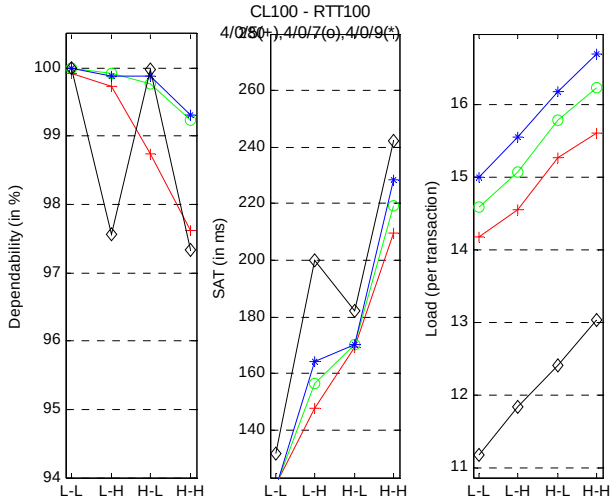


Fig.B.21, Regular, CL = 100s, RTT=100ms, FO=4, Retrans=0, comparing pool size

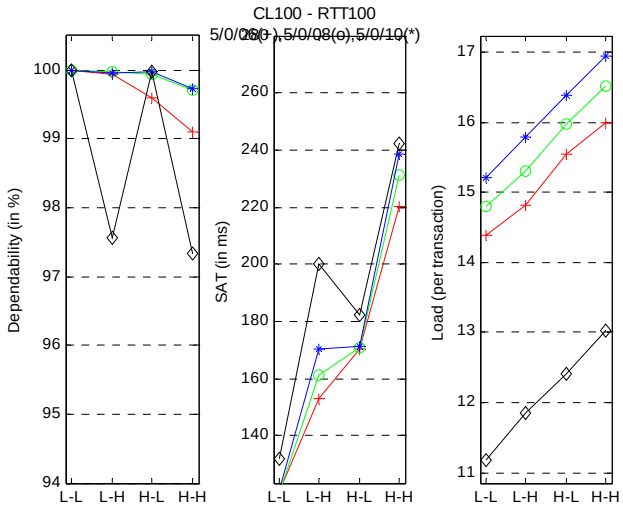


Fig.B.22, Regular, CL = 100s, RTT=100ms, FO=5, Retrans=0, comparing pool size

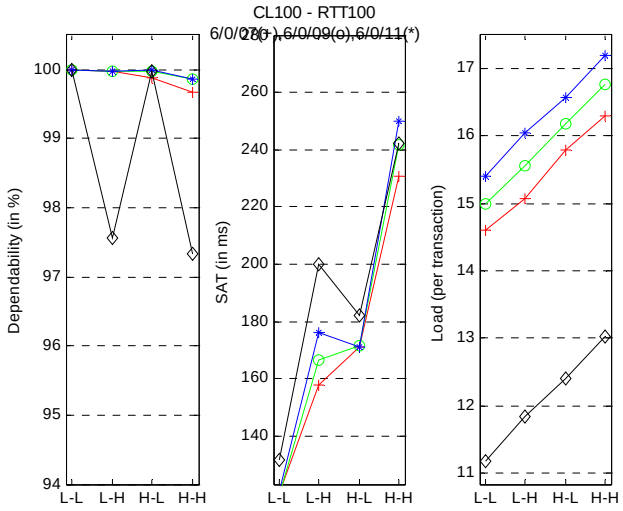


Fig.B.23, Regular, CL = 100s, RTT=100ms, FO=6, Retrans=0, comparing pool size

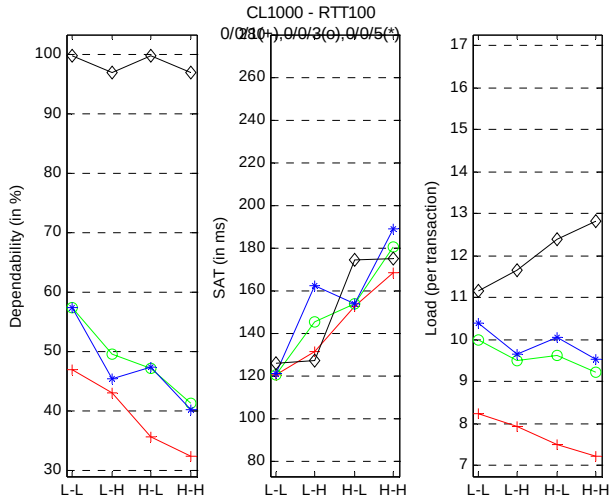


Fig.B.24, Regular, CL = 1000s, RTT=100ms, FO=0, Retrans=0, comparing pool size

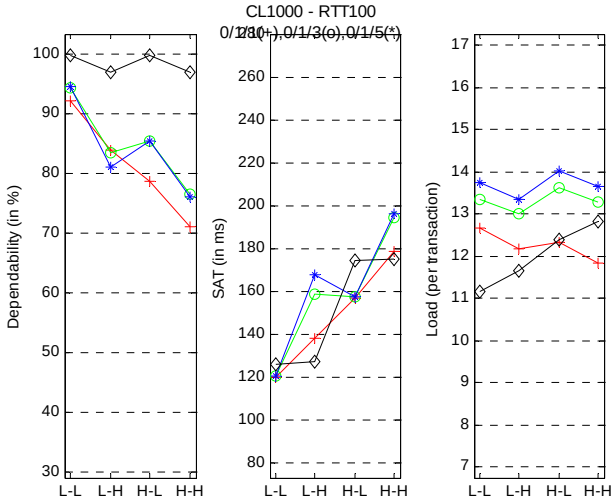


Fig.B.25, Regular, CL = 1000s, RTT=100ms, FO=0, Retrans=1, comparing pool size

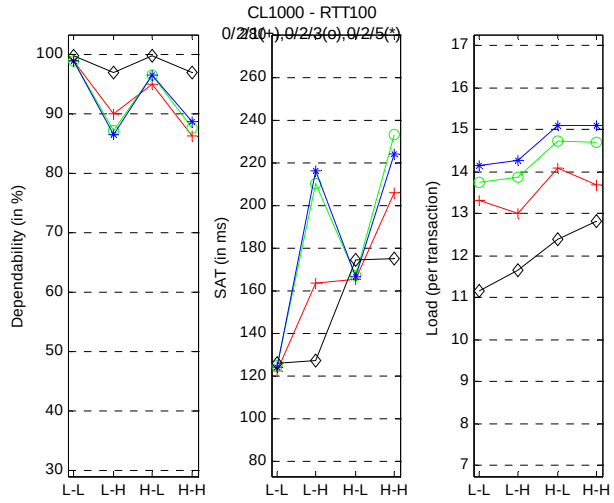


Fig.B.26, Regular, CL = 1000s, RTT=100ms, FO=0, Retrans=2, comparing pool size

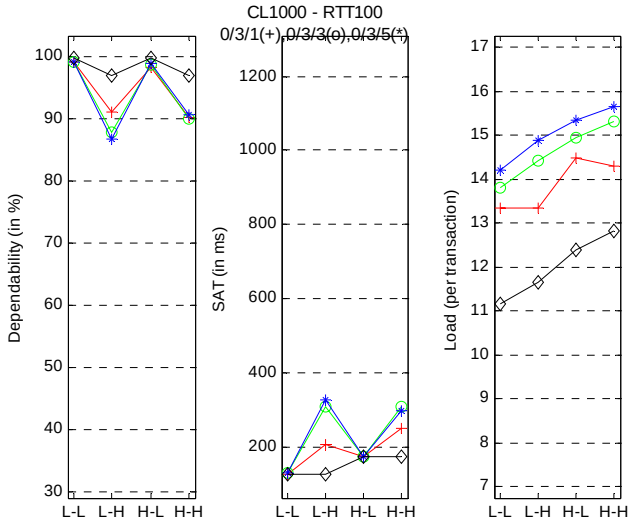


Fig.B.27, Regular, CL = 1000s, RTT=100ms, FO=0, Retrans=3, comparing pool size

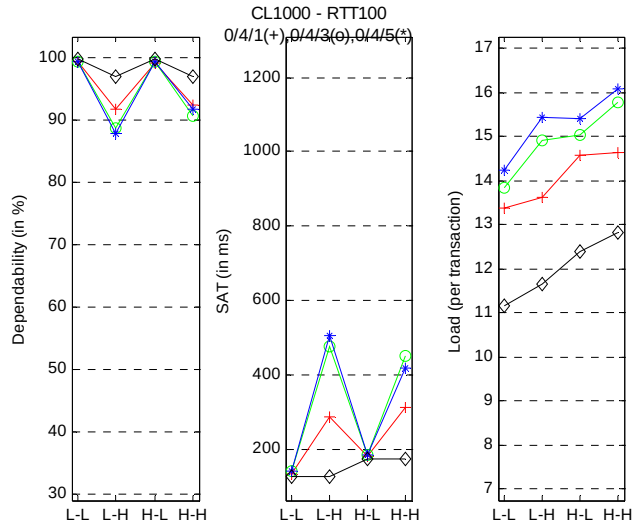


Fig.B.28, Regular, CL = 1000s, RTT=100ms, FO=0, Retrans=4, comparing pool size

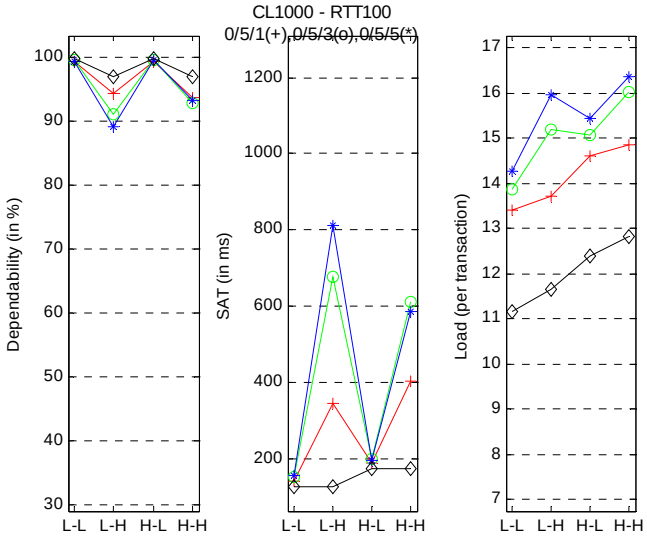


Fig.B.29, Regular, CL = 1000s, RTT=100ms, FO=0, Retrans=5, comparing pool size

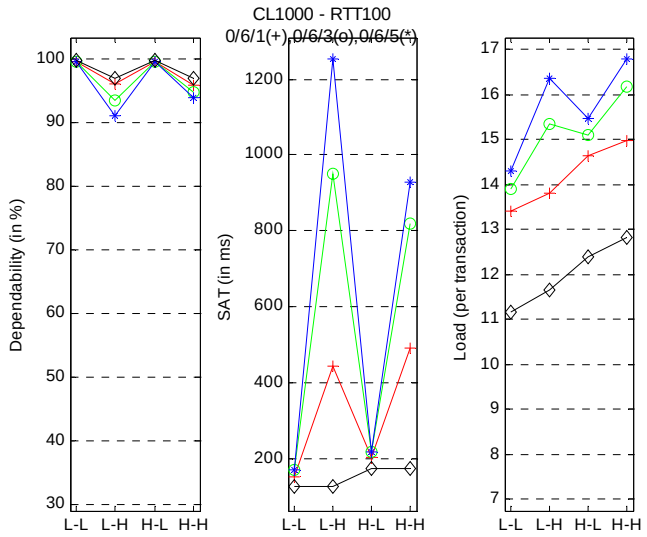


Fig.B.30, Regular, CL = 1000s, RTT=100ms, FO=0, Retrans=6, comparing pool size

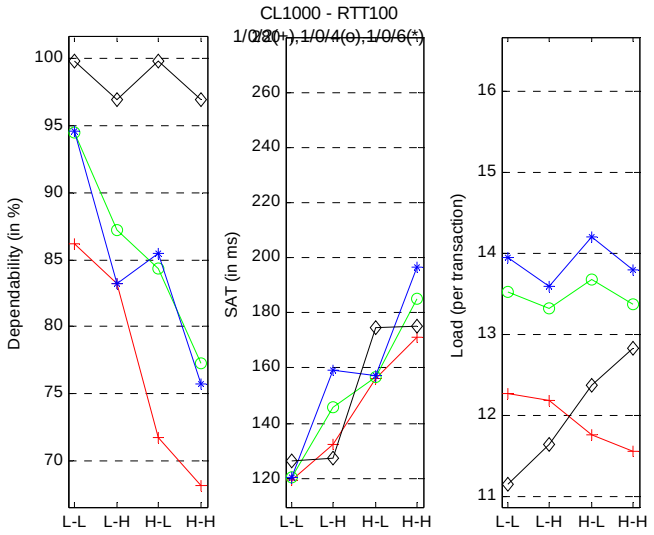


Fig.B.31, Regular, CL = 1000s, RTT=100ms, FO=1, Retrans=0, comparing pool size

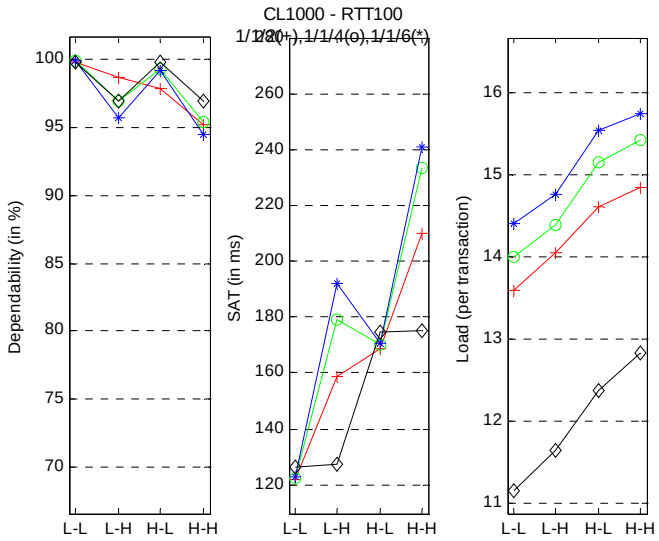


Fig.B.32, Regular, CL = 1000s, RTT=100ms, FO=1, Retrans=1, comparing pool size

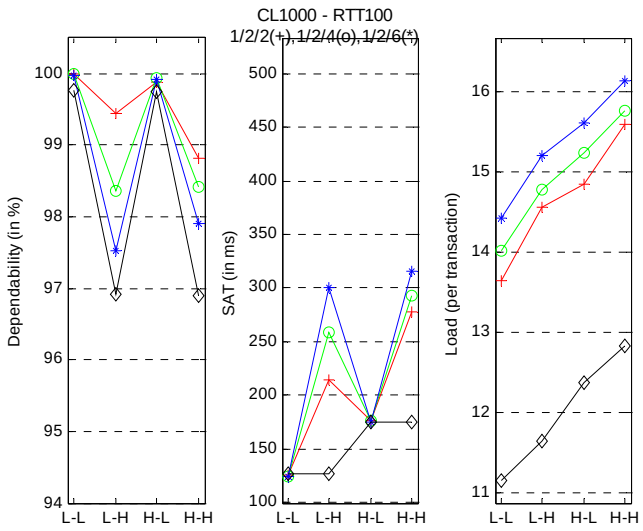


Fig.B.33, Regular, CL = 1000s, RTT=100ms, FO=1, Retrans=2, comparing pool size

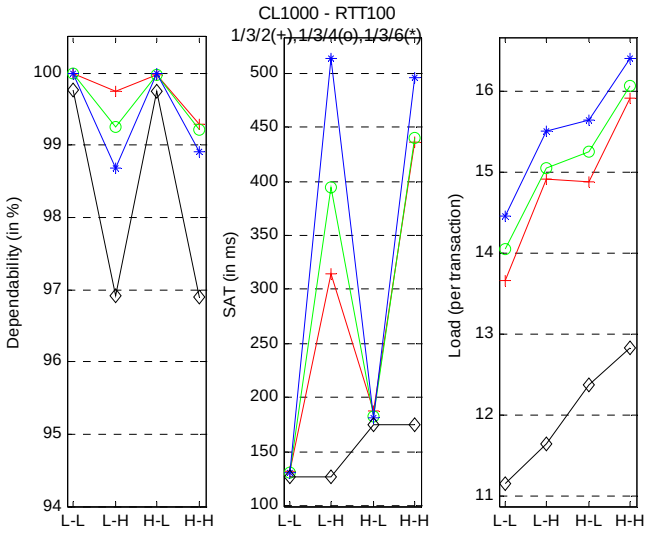


Fig.B.34, Regular, CL = 1000s, RTT=100ms, FO=1, Retrans=3 comparing pool size

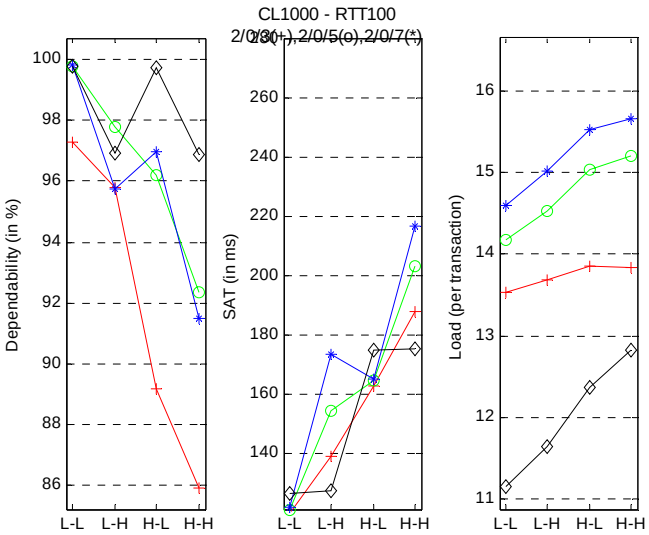


Fig.B.35, Regular, CL = 1000s, RTT=100ms, FO=2, Retrans=0, comparing pool size

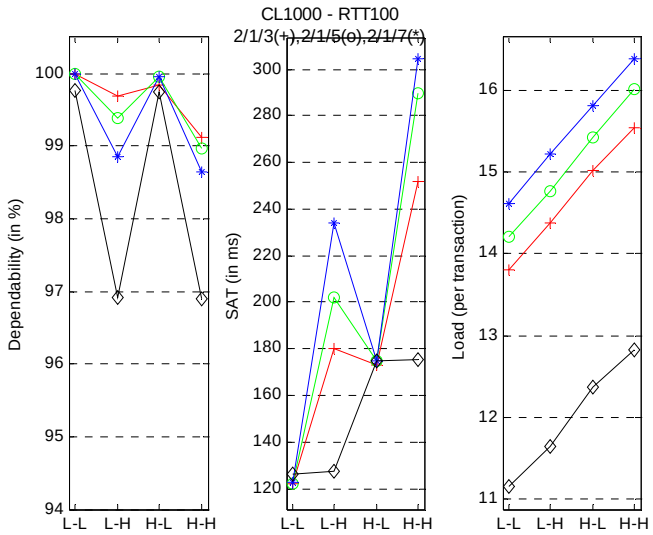


Fig.B.36, Regular, CL = 1000s, RTT=100ms, FO=2, Retrans=1, comparing pool size

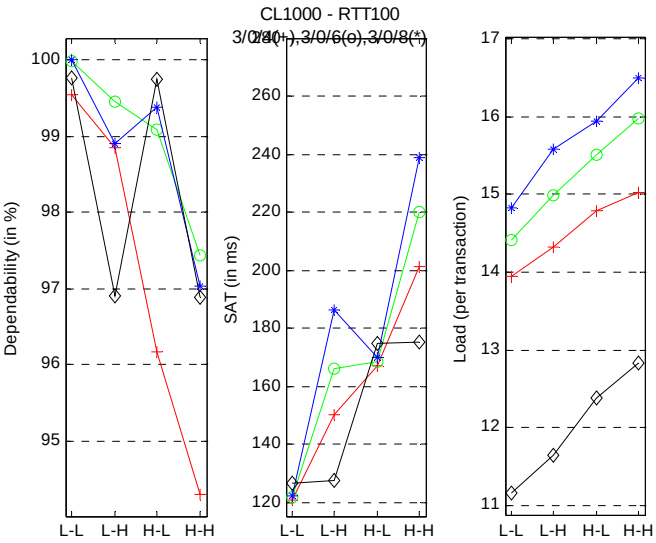


Fig.B.37, Regular, CL = 1000s, RTT=100ms, FO=3, Retrans=0, comparing pool size

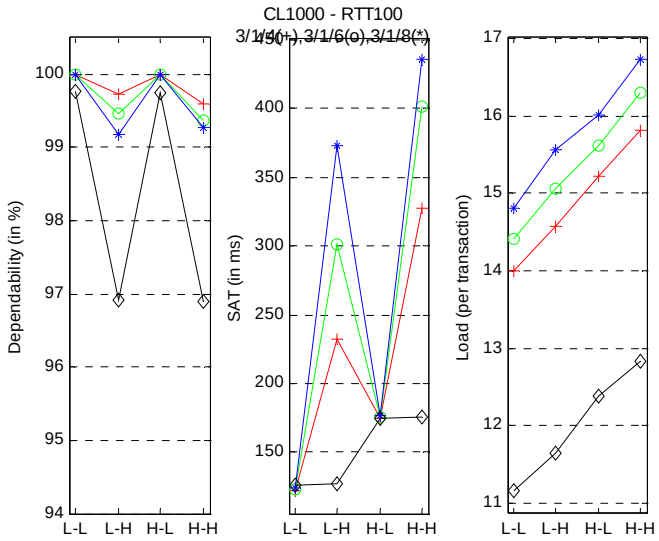


Fig.B.38, Regular, CL = 1000s, RTT=100ms, FO=3, Retrans=1, comparing pool size

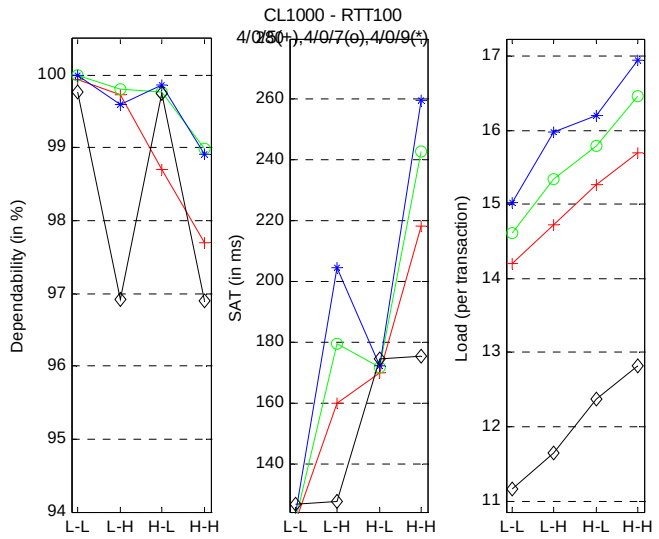


Fig.B.39, Regular, CL = 1000s, RTT=100ms, FO=4, Retrans=0, comparing pool size

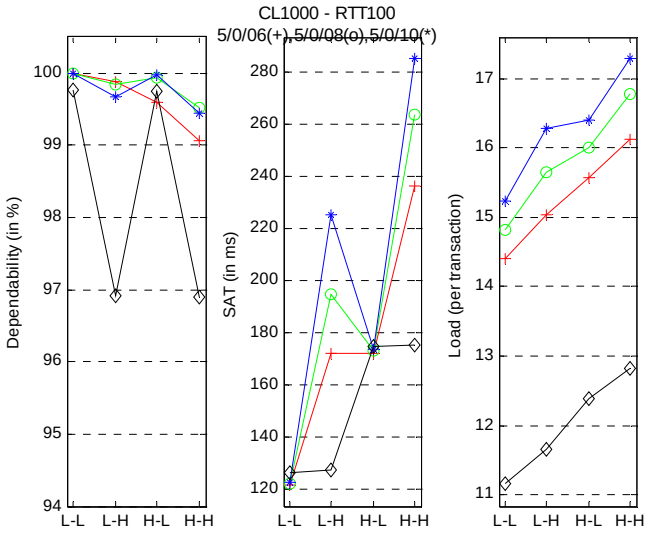


Fig.B.40, Regular, CL = 1000s, RTT=100ms, FO=5, Retrans=0, comparing pool size

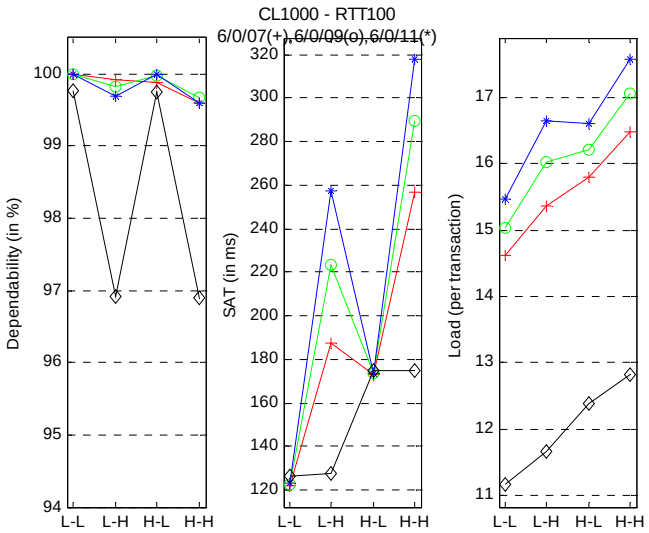


Fig.B.41, Regular, CL = 1000s, RTT=100ms, FO=6, Retrans=0, comparing pool size

References

[3GPP00] 3GPP, "3G security; security architecture", Tech. Rep. TS33.102, December 2000

[3GPP02a] 3GPP, "IP Multimedia Subsystem (IMS) (Release 5)", Tech. Rep. TS23.228, March 2002

[3GPP02b] 3GPP, "End-to-end Quality of Service (QoS) concept and architecture (Release 5)," Tech. Rep. TS23.207, March 2002

[3GPP02c] 3GPP, "Quality of Service (QoS) concept and architecture (Release 5)", Tech. Rep. TS23.107, March 2002

[3GPP04a] 3GPP, "Network architecture (Release 6)," Tech. Rep. TS23.002, December 2004

[3GPP04b] 3GPP, "General Packet Radio Service (GPRS), Service Decription (Release 6)" Tech. Rep. TS23.060, December 2004

[Avizienis04] A. Avizienis, J.C. Laprie, B. Randell, C. Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing," IEEE Transactions on Dependable and Secure Computing, vol.1, no.1, pp.11-33, January 2004

[Bluetothweb] http://www.bluetooth.com/Bluetooth/Technology/Building/Specifications/Default.htm

[Bozinovski02] M. Bozinovski, L. Gavriloska, R. Prasad, "Performance Evaluation of a SIP-based State-sharing Mechanism", 56th Vehicular Technology Conference (VTC), September 2002

[Bozinovski04a] M. Bozinovski, H-P. Schwefel, R. Prasad, "Algorithm for Controlling Transaction Consistency in SIP Session Control Systems", IEE Electronic Letters, vol.40, no.3, pp.209-211, February 2004

[Bozinovski04b] M. Bozinovski, *Algorithm for Controlling Transaction Consistency in SIP Session Control Systems*, Ph.D. dissertation, Aalborg University, Denmark, June 2004

[Braden97] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification," IETF, RFC2205, September 1997

[Bugnalo07] M. Bugnalo, "Hash Based Addresses (HBA)", IETF, draft-ietf-shim-hba-05, December2007 *(work in progress)*

[Castro06] G. Castro, K. Larsen, H-P. Schwefel, "Quality of Service Provisioning for Macro-Mobility in IMS-based Networks," 9th International Symposium on Wireless Personal Multimedia Communications (WPMC 06), September 2006

[Chen07] X. Chen, J. Rinne, J. Wiljakka, "Problem Statement for MIPv6 Interactions with GPRS/UMTS Packet Filtering," IETF, draft-chen-mip6-gprs-07.txt, January 2007 *(work in progress)*

[Degermark99] M. Degermark, B. Nordgren, S. Pink, "IP Header Compression," IETF, RFC2507, February 1999

[Dreibholz03] T. Dreibholz, A. Jungmaier, M. Tuexen, "A New Scheme for IP-based Internet Mobility," 28th IEEE Local Computer Networks Conference (LCNC 03), November 2003

[Durham00] D. Durham (Ed.) et al., "The COPS (Common Open Policy Service) Protocol," IETF, RFC2748, January 2000

[Faccin04] S. Faccin, P. Lalwaney, B. Patil, "IP Multimedia Services: Analysis of Mobile IP and SIP Interactions in 3G Networks," IEEE Communications Magazine, pp.113-120, January. 2004

[Faizan04] J. Faizan, H. El-Rewini, M. Khalil, "Problem Statement: Home Agent Reliability," IETF, draft-jfaizan-mipv6-ha-reliability-01.txt, February, 2004

[Fathi06] H. Fathi, *Real-Time Services in IP-based Wireless Heterogeneous Networks*, Ph.D. dissertation, Aalborg University, Denmark, March 2006

[Feng04] F. Feng, D. S. Reeves, "Explicit Proactive Handoff with Motion Prediction for Mobile IP," IEEE Wireless Communications and Networking Conference (WCNC 2004), March 2004

[FSC03] Fujitsu Siemens Computers, "RTP Overview and Programmer's Guide", Resilient Telco Platform V2.0, March 2003

[Groenbaek07] J. Grønbæk, H.P. Schwefel, T. Renier, H.P. Frejek, "Client-Centric Performance Analysis of a High-Availability Cluster," 4th International Service Availability Symposium (ISAS 2007), 2007

[Hamer03] L-N. Hamer, B. Gage, H. Shieh, "Framework for Session Set-up with Media Authorization", IETF, RFC3521, April 2003

[Handley98] M. Handley, V.Jacobson, "SDP: Session Description Protocol," IETF, RFC2327, April 1998

[Heddaya96] A. Heddaya, A. Helal, "Reliability, Availability, Dependability and Performability: A User-centered View," Tech. Rep. BU-CS-97-011, Boston University, December 1996

[Helal96] A. Helal, A. Heddaya, B. Bhargava, Replication Techniques in Distributed Systems. Kluwer Academic Publishers, 1996.

[Herlihy90] M. Herlihy, J. Wing, "Linearizability: a Correctness Condition for Concurrent Obejcts," ACM Transactions fo Programming Languages and Systems, vol.12, no.3, pp.463-492, July 1990

[IEEE_web] http://www.ieee802.org/11/

[ITU93] ITU-T, "Introduction to CCITT Signalling System No. 7", Recommendation Q.700, March 1993

[Johnson04] D. Jonhson, C. Perkins, J. Arkko, "Mobility Support in IPv6," IETF, RFC3775, June 2004

[Kim03] P. Kim, W. Boehm, "Support for Real-Time Applications in Future Mobile Networks: the IMS Approach," 6th Wireless Personal Multimedia Communications (WPMC 03), October 2003

[Kohler06a] E. Kohler, M. Handley, S. Floyd, "Datagram Congestion Control Protocol (DCCP)," IETF, RFC4340, March 2006

[Kohler06b] E. Kohler, "Generalized Connections in the Datagram Congestion Control Protocol," IETF, draft-kohler-dccp-mobility-02.txt, June 2006 *(work in progress)*

[Kwon02] T.T. Kwon, M. Gerla, Sa. Das, Su. Das, "Mobility Management for VoIP Services: Mobile IP vs. SIP," IEEE Wireless Communications, pp.66-75, October 2002

[Lamport78] L. Lamport, "Time, Clocks and the Ordering of Events in a Distributed System," Communications of the ACM, vol.21, no.7, pp.558-565, July 1978

[Larsen06a] K. Larsen, E. Matthiesen, H-P. Schwefel, G. Kuhn, "Optimized Macro Mobility within the 3GPP IP Multimedia Subsystem," IARIA International Conference on Wireless and Mobile Communications (ICWMC 06), July 2006

[Larsen06b] K. Larsen, H-P. Schwefel, G. Kuhn, "Migration of the Security Association for Fast SIP Mobility within the IP Multimedia Subsystem," 9th International Symposium on Wireless Personal Multimedia Communications (WPMC 06), September 2006

[Lei07] P. Lei, L.Ong, M. Tuexen, "*An Overview of Reliable Server Pooling Protocols*", IETF, draft-ietf-rserpool-overview-01.txt, April 2007

[Liu07] Y. Liu, *Virtual Backbone and Mobility-based Optimization in Wireless Multi-hop Networks*, Ph.D. dissertation, Aalborg University, Denmark, September 2007

[MAGNET_web] http://www.telecom.ntua.gr/magnet/index.html

[Meyer85] J. F. Meyer, A. Movaghar, and W. H. Sanders, "Stochastic activity networks: Structure, behavior, and application," International Workshop on Timed Petri Nets, pp.106–115, July 1985

[Möbius07] "Möbius, Model-Based Environment for Validation of System Reliability, Availability, Security and Performance—User Manual," 2007

[Moskovitz07] R. Moskovtz, P. Nikander, T. Henderson, "Host Identity Protocol", IETF, draft-ietf-hip-base-10, October 2007 *(work in progress)*

[O'Reagan04] E. O'Reagan, D. Pesch, "Performance Estimation of a SIP based Push-to-Talk Service for 3G Networks," 5th European Wireless Conference – Mobile and Wireless Systems beyond 3G, February 2004

[Perkins01] C. Perkins, D. Johnson, "Route Optimization in Mobile IP," IETF, draft-ietf-mobileip-optim-11.txt, September 2001 *(work in progress)*

[Perkins02] C. Perkins, "IP Mobility Support for IPv4," IETF, RFC3344, August 2002

[Peterson71] W. Peterson, E. Weldon, *Error Correcting Codes, Revised 2nd Edition*., MIT Press, 1971

[Postel82] J. Postel, "Simple Mail Transfer Protocol", IETF, RFC821, August 1982

[Postel85] J. Postel, J. Reynolds, "File Transfer Protocol (FTP)", IETF, RFC959, October 1985

[Prasad06] R. Prasad, L. Deneire, *From WPANs to Personal Networks – Technologies and Applications*, Artech House, 2006

[Renier06] T. Renier, E. Matthiesen, H.P. Schwefel, R. Prasad, "Inconsistency Evaluation in a Replicated IP-based Call Control System," 3rd International Service Availability Symposium (ISAS 06), May 2006

[Riegel06] M. Riegel, M. Tuexen, "Mobile SCTP," IETF, draft-riegel-tuexen-mobile-sctp-07.txt, October 2006 *(work in progress)*

[Roos03] A. Roos, M. Hartman, S. Dutnall, "Critical Issues for Roaming in 3G," IEEE Wireless Communications, pp.29-35, February 2003

[Rosen01] E. Rosen, A. Viswanathan, R. Callon, "Multiprotocol Label Switching Architecture," RFC3031, IETF, January 2001

[Rosenberg02] J. Rosenberg et al., "SIP: Session Initiation Protocol," IETF, RFC3261, June 2002

[Shand07] M. Shand, S. Bryant, "IP Fast-reroute Framework," IETF, draft-ietf-rtgwg-ipfrr-framework-07.txt, June 2007 *(work in progress)*

[Schulzrinne00] H. Schulzrinne, E. Wedlund, "Application-Layer Mobility Using SIP," ACM Mobile Computing and Communications Review, vol.4, no.3, pp.47-57, July 2000

[Schulzrinne03] H. Schulzrinne, S. Casner, R. ,Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," IETF, RFC3550, July 2003 (*includes RTCP specifications)*

[Srivastava05] V. Srivastava, M. Motani, "Cross-Layer Design: A Survey and the Road Ahead," IEEE Communications, pp.112-119, December 2005

[Stallings96] W. Stallings, "IPv6: The New Internet Protocol," IEEE Communications Magazine, pp.96-108, July 1996

[Stewart00] R. Stewart, et al., "Stream Control Transmission Protocol," IETF, RFC2960, October 2000

[Stewart06a] R. Stewart, et al., "Aggregate Server Access Protocol (ASAP)," IETF, draft-ietf-rserpool-asap-11.txt, June 2006 *(work in progress)*

[Stewart06b] R. Stewart, et al., "Endpoint Handlespace Redundancy Protocol (ENRP)," IETF, draft-ietf-rserpool-enrp-11.txt, June 2006 *(work in progress)*

[Tanenbaum02] A. S. Tanenbaum, M. van Steen, *Distributed Systems—Principles and Paradigms*, pp.291-360, Practice Hall, 2002

[Tuexen02] M. Tuexen, et al., "Requirements for Reliable Server Pooling," IETF, RFC3237, January 2002

[vanMoorsel06] A. van Moorsel, K. Wolter, "Analysis of Restart Mechanisms in Software Systems," IEEE Transactions on Software Engineering, vol.32, no.8, pp.547-558, August 2006

[Wimax_web] http://www.wimaxforum.org/technology/documents/

[Yu00] H. Yu, A. Vahdat, "Design and Evaluation of a Continuous Consistency Model for Replicated Services," 4th Symposium on Operating Systems Design and Implementation (OSDI 00), October 2000

[Zhang06] Y. Zhang, M. Fujise, "Security Management in the Next Generation Wireless Networks," International Journal of Network Security, vol.3, no.1, pp.1-7, July 2006

[Åhlund03] C. Åhlumd, A. Zaslavsky, "*Multihoming with Mobile IP*", 6th IEEE Conference on High Speed Networks and Multimedia Communications (HSNMC 03), July 2003

Author's Publications

Peer Reviewed Conference Papers

T. Renier, H. Fathi, H.P. Schwefel, R. Prasad, "MIP-based enhanced mid-session macro-mobility for IMS-controlled stateful applications," 10th International Symposium on Wireless Personal Multimedia Communications (WPMC 2007), December 2007

T. Renier, E. Matthiesen, H.P. Schwefel, "Inconsistency Evaluation in a Replicated IP-based Call Control System," 3rd International Service Availability Symposium, (ISAS 2006), May 2006

T. Renier, H. Fathi, G. Kuhn, H.P. Schwefel, "MIPv6 Operations in IMS-based Access Networks," 9th International Symposium on Wireless Personal Multimedia Communications (WPMC 2006), September 2006

T. Renier, M. Bozinovski, K. Larsen, H.P. Schwefel, R. Prasad, R. Seidl, "Distributed redundancy or cluster solution? An experimental evaluation of two approaches for dependable mobile Internet services," 1st International Service Availability Symposium (ISAS 2004), May 2004

A. Nickelsen, J. Grønbæk, T. Renier, H.P. Schwefel, "Probabilistic Fault-Diagnostic in Mobile Networks Using Cross-Layer Observations," *to be submitted…*

E. Matthiesen, H.P. Schwefel, T. Renier, "A Selection Metric for Backup Group Creation in Inter-Vehicular Networks : A step in the way of distributed ad-hoc group state sharing," 16th IST Mobile and Wireless Communications Summit, 2007 (poster)

J. Grønbæk, H.P. Schwefel, T. Renier, H.P. Frejek, "Client-Centric Performance Analysis of a High-Availability Cluster," 4th International Service Availability Symposium (ISAS 2007), 2007

E. Matthiesen, T. Renier, M. Bozinovski, H.P. Schwefel, "Adaptive Partitioning Algorithms for Optimized State Replication of Highly Available Services," 8th International Symposium on Wireless Personal Multimedia Communications (WPMC 2005), 2005

M. Bozinovski, T. Renier, H.P. Schwefel, R. Prasad, "Adaptive, Scalable Framework for Dependable Peer-to-Peer Distributed Computing," 8th International Symposiu on Wireless Personal Multimedia Communications (WPMC 2005), 2005

M. Bozinovski, T. Renier, H.P. Schwefel, R. Prasad, "Transaction Consistency in Replicated SIP Call Control Systems," 4th International Conference on Information,

Communications & Signal Processing and Fourth Pacific-Rim Conference on Multimedia (ICICS-PCM 2003), 2003

P. Popovski, L. Gravilovska, T. Renier, H. Fathi, R. Prasad, "Energy-efficient Interference Avoidance for Interconnected Bluetooth Personal Area Networks," 57th IEEE Semiannual Vehicular Technology Conference (VTC 2003), 2003

Magazine Paper

T. Renier, K. Larsen, G. Castro, H.P. Schwefel, "Mid-Session Macro-Mobility in IMS-based Networks," IEEE Vehicular Technology Magazine, special issue on "IMS as service delivery platform for converged networks," Vol. 2, No. 1, March 2007

Technical Reports

"HIDENETS D4.1.2: Identification and development of evaluation methodologies, techniques and tools," Institut for Elektroniske Systemer, Aalborg Universitet, 2008

"HIDENETS D4.2.1: Application of the evaluation framework to the complete scenario (preliminary version)," Institut for Elektroniske Systemer, Aalborg Universitet, 2007

"HIDENETS D1.2: Revised reference model," Institut for Elektroniske Systemer, Aalborg Universitet, 2007

"HIDENETS D2.1.2: Resilient architecture," Institut for Elektroniske Systemer, Aalborg Universitet, 2006

"Mobility Schemes for Future Mobile Network - SIP mobility within the IMS," Siemens deliverable, 2005

"RTP integration into 3GPP architecture, evaluation and comparison of RSerPool and RTP, and dynamic server selection policy," Siemens deliverable, 2004

"Migration to 3GPP architecture (SIP007++) and concepts of RTP integration and heart-beat mechanisms," Siemens deliverable, 2003

"Integration of SIP into RSerPool and implementation of an extended SIP call scenario," Siemens deliverable, 2002